```
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
```

FILEID**FORNMLTAB

```
FFFFFFFFFF    000000    RRRRRRRR    NN      NN  MM      MM  LL          TTTTTTTTTT    AAAAAA    BBBBBBBB
FFFFFFFFFF    000000    RRRRRRRR    NN      NN  MM      MM  LL          TTTTTTTTTT    AAAAAA    BBBBBBBB
FF          00      00  RR      RR  NN      NN  MMMM  MMMM  LL              TT      AA      AA  BB      BB
FF          00      00  RR      RR  NN      NN  MMMM  MMMM  LL              TT      AA      AA  BB      BB
FF          00      00  RR      RR  NNNN    NN  MM  MM  MM  LL              TT      AA      AA  BB      BB
FF          00      00  RR      RR  NNNN    NN  MM  MM  MM  LL              TT      AA      AA  BB      BB
FFFFFFFF    00      00  RRRRRRRR    NN  NN  NN  MM      MM  LL              TT      AA      AA  BBBBBBBB
FFFFFFFF    00      00  RRRRRRRR    NN  NN  NN  MM      MM  LL              TT      AA      AA  BBBBBBBB
FF          00      00  RR  RR      NN    NNNN  MM      MM  LL              TT      AAAAAAAAAA  BB      BB
FF          00      00  RR  RR      NN    NNNN  MM      MM  LL              TT      AAAAAAAAAA  BB      BB
FF          00      00  RR    RR    NN      NN  MM      MM  LL              TT      AA      AA  BB      BB    ....
FF          00      00  RR    RR    NN      NN  MM      MM  LL              TT      AA      AA  BB      BB    ....
FF            000000    RR      RR  NN      NN  MM      MM  LLLLLLLLLL      TT      AA      AA  BBBBBBBB    ....
FF            000000    RR      RR  NN      NN  MM      MM  LLLLLLLLLL      TT      AA      AA  BBBBBBBB    ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE FOR$$NML_TABLES (%TITLE 'FOR$$NML_TABLES - TPARSE state tables for NAMELIST input'
;     2          0002  0                       IDENT = '1-012'           ! File: FORNMLTAB.B32 Edit: SBL1012
;     3          0003  0                       ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                      *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    17          0017  1 !*  TRANSFERRED.                                                              *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    21          0021  1 !*  CORPORATION.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:     FORTRAN Language Support
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains the LIB$TPARSE state tables used in
;    36          0036  1 !       implementing FORTRAN NAMELIST input.  It also contains the
;    37          0037  1 !       action routines associated with the state tables.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:  User mode - AST reentrant
;    40          0040  1 !
;    41          0041  1 ! AUTHOR: Steven B. Lionel, CREATION DATE: 10-July-1980
;    42          0042  1 !
;    43          0043  1 ! MODIFIED BY:
;    44          0044  1 !
;    45          0045  1 ! 1-001 - Original.  SBL 10-July-1980
;    46          0046  1 ! 1-002 - Disallow superfluous commas.  SBL 18-Nov-1980
;    47          0047  1 ! 1-003 - Reflect change in group block spec so that number-of-variables is
;    48          0048  1 !         a word; second word is reserved.  SBL 5-Dec-1980
;    49          0049  1 ! 1-004 - Don't require a delimiter before ending $ or &.  These characters can
;    50          0050  1 !         no longer be a part of a logical constant.  SBL 17-Dec-1980
;    51          0051  1 ! 1-005 - Allow repeated nulls of the form "r*".  Don't consider repeated values
;    52          0052  1 !         as candidates for being identifiers.  Add comments.  SBL 2-Mar-1981
;    53          0053  1 ! 1-006 - Add text describing the NAMELIST descriptor block.  Disallow an array
;    54          0054  1 !         substring without a subscript.  SBL 15-April-1981
;    55          0055  1 ! 1-007 - Change to use new OTS$CVT_T_F routine.  SBL 15-April-1981
;    56          0056  1 ! 1-008 - Also use OTS$CVT_T_F in STORE_COMPLEX.  SBL 5-June-1981
;    57          0057  1 ! 1-009 - Use new ONE_OF macro where necessary.  SBL 18-Dec-1981
```

```
;    58          0058  1 ! **** Start post-V3.0 enhancements. ****
;    59          0059  1 ! 1-010 - Enhancements and minor bug fixes:  SBL 17-Dec-1982
;    60          0060  1 !               1. Allow "!" to begin an end-of-line comment.  It is allowed
;    61          0061  1 !                  wherever "end-of-line" is allowed, except in character values,
;    62          0062  1 !                  and is equivalent to end-of-line.
;    63          0063  1 !               2. Disallow signed integer as syntactically correct for
;    64          0064  1 !                  repeat count.
;    65          0065  1 !               3. Improve error reporting by chaining FOR$_INVTEXREC for
;    66          0066  1 !                  input conversion errors.
;    67          0067  1 !               4. Use prologue file.
;    68          0068  1 ! 1-011 - Turn off NML$V_IMAG in STORE_VALUE so that subsequent complex values
;    69          0069  1 !             get stored correctly.  (Same as BUG 1-009A)  SPR 11-nnnnn SBL 7-Mar-1983
;    70          0070  1 ! 1-012 - Add inquiry feature.  SBL 24-May-1983
;    71          0071  1 !--
;    72          0072  1
```

```
;    74          0073  1 %SBTTL 'Declarations'
;    75          0074  1 !
;    76          0075  1 ! PROLOGUE FILE:
;    77          0076  1 !
;    78          0077  1
;    79          0078  1 REQUIRE 'RTLIN:FORPROLOG';                      ! FORTRAN-specific declarations
;    80          0144  1
;    81          0145  1 !
;    82          0146  1 ! LINKAGES:
;    83          0147  1 !
;    84          0148  1
;    85          0149  1 LINKAGE
;    86          0150  1     JSB_COMPARE_UPCASE = JSB (REGISTER=4, REGISTER=5) :
;    87          0151  1                          NOPRESERVE (0,1,2,3,4) NOTUSED (6,7,8,9,10,11);
;    88          0152  1
;    89          0153  1 !
;    90          0154  1 ! TABLE OF CONTENTS:
;    91          0155  1 !
;    92          0156  1
;    93          0157  1 FORWARD ROUTINE
;    94          0158  1     NEXT_RECORD,                          ! Read another record
;    95          0159  1     LOOKUP_IDENTIFIER,                    ! Lookup identifier
;    96          0160  1     SUBSTRING_COLON,                      ! Process colon in substring
;    97          0161  1     INIT_SUBS,                            ! Start a subscript/substring
;    98          0162  1     STORE_SUBS,                           ! Store a subscript/substring
;    99          0163  1     END_SUBSCRIPT,                        ! End a subscript
;   100          0164  1     END_SUBSTRING,                        ! End a substring
;   101          0165  1     CONVERT_INTEGER,                      ! Convert a decimal integer
;   102          0166  1     STORE_LOGICAL,                        ! Store a logical into CONSBLOCK
;   103          0167  1     STORE_REAL,                           ! Store a real value into CONSBLOCK
;   104          0168  1     STORE_COMPLEX,                        ! Store a complex value into CONSBLOCK
;   105          0169  1     STORE_REPEAT,                         ! Store repeat count
;   106          0170  1     END_REPEAT,                           ! End a repeated value
;   107          0171  1     STORE_CHARACTER,                      ! Store a character string character
;   108          0172  1     END_CHARACTER,                        ! End a character string
;   109          0173  1     STRING_OK,                            ! Is a string value ok?
;   110          0174  1     STORE_VALUE,                          ! Store a value
;   111          0175  1     NULL_VALUE,                           ! Skip an element
;   112          0176  1     SET_VALUE_IDENT,                      ! Indicate last value was an identifier
;   113          0177  1     WAS_VALUE_IDENT,                      ! Lookup last value token as an identifier
;   114          0178  1     SYNTAX_ERROR,                         ! Signal a syntax error
;   115          0179  1     INVREFVAR_ERROR,                      ! Signal invalid ref to variable error
;   116          0180  1     INPCONERR_ERROR,                      ! Signal input conversion error
;   117          0181  1     BLANKS_OFF,                           ! Turn explicit blanks off
;   118          0182  1     BLANKS_ON,                            ! Turn explicit blanks on
;   119          0183  1     COMPUTE_INDEX,                        ! Compute the subscript index
;   120          0184  1     COMPARE_UPCASE: JSB_COMPARE_UPCASE,   ! Compare strings upcased
;   121          0185  1     DUMP_NAMES,                           ! Respond to '?' inquiry
;   122          0186  1     DUMP_VALUES;                          ! Respond to '=?' inquiry
;   123          0187  1
;   124          0188  1 !
;   125          0189  1 ! REQUIRE FILES:
;   126          0190  1 !
;   127          0191  1
;   128          0192  1 LIBRARY 'RTLTPAMAC';                          ! TPARSE library of macros
;   129          0193  1
;   130          0194  1 !
```

```
;  131          0195  1 ! EQUATED SYMBOLS:
;  132          0196  1 !
;  133          0197  1
;  134          0198  1 LITERAL
;  135          0199  1     SINGLE_QUOTE = 39,                  ! ASCII value for ''''
;  136          0200  1     K_NULL = 0,                         ! Constant type for null value
;  137          0201  1     K_LOGICAL = 1,                      ! Constant type for logical
;  138          0202  1     K_INTEGER = 2,                      ! Constant type for integer
;  139          0203  1     K_REAL = 3,                         ! Constant type for real
;  140          0204  1     K_COMPLEX = 4,                      ! Constant type for complex
;  141          0205  1     K_CHARACTER = 5;                    ! Constant type for character
;  142          0206  1
;  143          0207  1 !
;  144          0208  1 ! FIELDS:
;  145          0209  1 !
;  146          0210  1 !       NONE
;  147          0211  1 !
;  148          0212  1 ! OWN STORAGE:
;  149          0213  1 !
;  150          0214  1 !       NONE
;  151          0215  1 !
;  152          0216  1 ! BUILTIN DECLARATIONS:
;  153          0217  1 !
;  154          0218  1 BUILTIN
;  155          0219  1     CALLG,
;  156          0220  1     INDEX;
;  157          0221  1
;  158          0222  1 !
;  159          0223  1 ! EXTERNAL REFERENCES:
;  160          0224  1 !
;  161          0225  1
;  162          0226  1 EXTERNAL ROUTINE
;  163          0227  1     FOR$$CVT_TYPE,                      ! Convert a value to destination type
;  164          0228  1     FOR$$DO_NML_OUTPUT: CALL_CCB,       ! Do Namelist output
;  165          0229  1     FOR$$REC_RSNO: JSB_RECO,            ! Read a record
;  166          0230  1     FOR$$REC_WSNO: JSB_RECO,            ! Start a write
;  167          0231  1     FOR$$SIGNAL: NOVALUE,               ! Signal continuable error
;  168          0232  1     FOR$$SIGNAL_STO: NOVALUE,           ! Signal fatal error
;  169          0233  1     OTS$CVT_TI_L,                       ! Convert decimal to longword
;  170          0234  1     OTS$CVT_TL_L,                       ! Convert logical to longword
;  171          0235  1     OTS$CVT_T_F,                        ! Convert text to F_floating
;  172          0236  1     OTS$CVT_T_D,                        ! Convert text to D_floating
;  173          0237  1     OTS$CVT_T_G,                        ! Convert text to G_floating
;  174          0238  1     OTS$CVT_T_H,                        ! Convert text to H_floating
;  175          0239  1     LIB$SIG_TO_RET;                     ! Convert signal to return value
;  176          0240  1
;  177          0241  1 !<BLF/PAGE>
```

```
1 !++
1 !     Each NAMELIST descriptor block has the following form:
1 !
1 !             3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 !             1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
1 !
1 !            +---------------------------------------------------------------+
1 !         0  |            Address of ASCIC name of NAMELIST group            |
1 !            +---------------------------------------------------------------+
1 !         1  |          Reserved           |   Count of NAMELIST variables   |
1 !            +---------------------------------------------------------------+
1 !         2  |               Address of ASCIC name of variable 1             |
1 !            +---------------------------------------------------------------+
1 !         3  |        Address of standard VAX descriptor for variable 1      |
1 !            +---------------------------------------------------------------+
1 !         4  |                              ...                              |
1 !            +---------------------------------------------------------------+
1 !         5  |               Address of ASCIC name of variable n             |
1 !            +---------------------------------------------------------------+
1 !         6  |        Address of standard VAX descriptor for variable n      |
1 !            +---------------------------------------------------------------+
1 !
1 !
1 !     The NAMELIST group name and the variable names which are pointed to in
1 !     the  NAMELIST  descriptor  block  are  upper  case  only.  The FORTRAN
1 !     compiler or other calling program is responsible for  case  conversion
1 !     of the name strings.  In NAMELIST input data, case is significant only
1 !     in character literals.  The run-time library is responsible  for  case
1 !     conversion of NAMELIST input data.
1 !
1 !     The allowable data types in variable descriptors are  BU  (BYTE),  WU,
1 !     LU,  W,  L,  F,  D, G, H, T, FC, DC, and GC.  The allowable descriptor
1 !     classes are scalar and array.  For the  array  class  descriptor,  the
1 !     descriptor  flags  COLUMN,  COEFF,  and BOUNDS must be set, indicating
1 !     column-major order and the presence of coefficient and bounds  blocks.
1 !     The number of dimensions must not exceed 7.
1 !--
1
1 !<BLF/PAGE>
```

```
;  219          0281  1 %SBTTL 'FOR$$NML_TABLES - TPARSE tables for NAMELIST input'
;  220          0282  1
;  221          0283  1 !++
;  222          0284  1 ! FUNCTIONAL DESCRIPTION:
;  223          0285  1 !
;  224          0286  1 !       The following are the state tables used to perform FORTRAN
;  225          0287  1 !       NAMELIST input.
;  226          0288  1 !
;  227          0289  1 !--
;  228          0290  1
;  229          0291  1 $INIT_STATE (FOR$$A_NMLSTATE, FOR$$A_NMLKEYWD);
;  230          0292  1
;  231          0293  1 !+
;  232          0294  1 ! Main scanning loop.  Look for assignments.
;  233          0295  1 ! If a $ or & is found, terminate the statement.
;  234          0296  1 !-
;  235        P 0297  1 $STATE (BEGIN_SCAN,
;  236        P 0298  1     ((END_OF_LINE), BEGIN_SCAN, NEXT_RECORD),
;  237        P 0299  1     ('$', TPA$_EXIT),
;  238        P 0300  1     ('&', TPA$_EXIT),
;  239        P 0301  1     ((ASSIGNMENT), BEGIN_SCAN, BLANKS_OFF),
;  240        P 0302  1     (TPA$_LAMBDA, ERROR_STATE)
;  241          0303  1     );
;  242          0304  1
;  243          0305  1 !+
;  244          0306  1 ! This state matches the equivalent of an end-of-line; either the
;  245          0307  1 ! actual end-of-line or a comment beginning with "!", but it does
;  246          0308  1 ! not consume the "!".
;  247          0309  1 !-
;  248        P 0310  1 $STATE (END_OF_LINE,
;  249        P 0311  1     (TPA$_EOS, TPA$_EXIT),
;  250        P 0312  1     ((NO_COMMENT), TPA$_FAIL),
;  251        P 0313  1     (TPA$_LAMBDA, TPA$_EXIT)
;  252          0314  1     );
;  253          0315  1
;  254          0316  1 !+
;  255          0317  1 ! An assignment consists of a variable, an equals sign, and a list of values.
;  256          0318  1 !-
;  257        P 0319  1 $STATE (ASSIGNMENT,
;  258        P 0320  1     ((VARIABLE), ASSN_EQL ,BLANKS_OFF),
;  259        P 0321  1     ('?', FLUSH_RECORD, DUMP_NAMES),              ! Dump names
;  260        P 0322  1     ((EQUALS_QUESTION), FLUSH_RECORD, DUMP_VALUES), ! Dump values and retry
;  261        P 0323  1     (TPA$_LAMBDA, ERROR_STATE)
;  262          0324  1     );
;  263          0325  1
;  264        P 0326  1 $STATE (FLUSH_RECORD,
;  265        P 0327  1     (TPA$_EOS, TPA$_EXIT),
;  266        P 0328  1     (TPA$_ANY, FLUSH_RECORD)
;  267          0329  1     );
;  268          0330  1
;  269        P 0331  1 $STATE (ASSN_EQL,
;  270        P 0332  1     ((END_OF_LINE), ASSN_EQL, NEXT_RECORD),
;  271        P 0333  1     ('=', VALUE_LIST),
;  272        P 0334  1     (TPA$_LAMBDA, ERROR_STATE)
;  273          0335  1     );
;  274          0336  1
;  275          0337  1 !+
```

```
;  276          0338  1 ! A value list consists of simple values and repeated values, possibly separated
;  277          0339  1 ! by commas.  A comma instead of a value indicates an omitted value, where that
;  278          0340  1 ! element of the variable should remain unchanged.
;  279          0341  1 !-
;  280        P 0342  1 $STATE (VALUE_LIST,
;  281        P 0343  1     ((END_OF_LINE), VALUE_LIST, NEXT_RECORD),
;  282        P 0344  1     (',', VALUE_LIST, NULL_VALUE),
;  283        P 0345  1     ((REPEATED_VALUE), VALUE_LIST1, BLANKS_ON),
;  284        P 0346  1     ((VALUE), VALUE_LIST1, BLANKS_ON),
;  285        P 0347  1     (TPA$_LAMBDA, TPA$_EXIT)
;  286          0348  1     );
;  287          0349  1
;  288          0350  1 !+
;  289          0351  1 ! A value has been found.  The next delimiter tells us if that token was really
;  290          0352  1 ! a value or was an identifier that looked like a value.
;  291          0353  1 !-
;  292        P 0354  1 $STATE (VALUE_LIST1,
;  293        P 0355  1     ((END_OF_LINE), VALUE_LIST2, BLANKS_OFF),
;  294        P 0356  1     (TPA$_BLANK, VALUE_LIST2, BLANKS_OFF),
;  295        P 0357  1     ((NO_LPAREN), VALUE_LIST2, BLANKS_OFF),     ! Succeeds if "(" NOT found
;  296        P 0358  1     (TPA$_LAMBDA, TPA$_EXIT, SET_VALUE_IDENT)   ! Last token was an identifier
;  297          0359  1     );
;  298          0360  1
;  299          0361  1 !+
;  300          0362  1 ! At this point, the last token was an identifier only if the next significant
;  301          0363  1 ! character is an "=".  The other case, a "(", was taken care of in the
;  302          0364  1 ! previous state.
;  303          0365  1 !-
;  304        P 0366  1 $STATE (VALUE_LIST2,
;  305        P 0367  1     ((END_OF_LINE), VALUE_LIST2, NEXT_RECORD),
;  306        P 0368  1     (TPA$_BLANK, VALUE_LIST2),                  ! Even though explicit blank
;  307        P 0369  1                                                 ! processing is off, use up
;  308        P 0370  1                                                 ! blanks in the record to aid
;  309        P 0371  1                                                 ! error reporting.
;  310        P 0372  1     (',', VALUE_LIST, STORE_VALUE),
;  311        P 0373  1     ((NO_EQUALS), VALUE_LIST, STORE_VALUE),     ! Succeeds if "=" NOT found
;  312        P 0374  1     (TPA$_LAMBDA, TPA$_EXIT, SET_VALUE_IDENT)   ! Last token was an identifier
;  313          0375  1     );
;  314          0376  1
;  315          0377  1 !+
;  316          0378  1 ! This type of state determines if the next character is "(", without consuming
;  317          0379  1 ! the character.  In this case, failure indicates that the desired character
;  318          0380  1 ! was found.  This scheme is used in the next, and in other states.
;  319          0381  1 !-
;  320        P 0382  1 $STATE (NO_LPAREN,
;  321        P 0383  1     ('(', TPA$_FAIL),
;  322        P 0384  1     (TPA$_LAMBDA, TPA$_EXIT)
;  323          0385  1     );
;  324          0386  1
;  325        P 0387  1 $STATE (NO_EQUALS,
;  326        P 0388  1     ((NO_EQUALS_QUESTION), NO_EQUALS2),
;  327        P 0389  1     (TPA$_LAMBDA, TPA$_EXIT)
;  328          0390  1     );
;  329          0391  1
;  330        P 0392  1 $STATE (NO_EQUALS2,
;  331        P 0393  1     ('=', TPA$_FAIL),
;  332        P 0394  1     (TPA$_LAMBDA, TPA$_EXIT)
```

```
;  333          0395  1      );
;  334          0396  1
;  335        P 0397  1  $STATE (NO_EQUALS_QUESTION,
;  336        P 0398  1      ((EQUALS_QUESTION), TPA$_FAIL),
;  337        P 0399  1      (TPA$_LAMBDA, TPA$_EXIT)
;  338          0400  1      );
;  339          0401  1
;  340          0402  1  !+
;  341          0403  1  ! Look for '=?'
;  342          0404  1  !-
;  343        P 0405  1  $STATE (EQUALS_QUESTION,
;  344        P 0406  1      ('=', , BLANKS_ON)                  ! Does it start with '='?
;  345          0407  1      );
;  346          0408  1
;  347        P 0409  1  $STATE (,
;  348        P 0410  1      ('?', TPA$_EXIT, BLANKS_OFF),       ! '=?' found
;  349        P 0411  1      (TPA$_LAMBDA, TPA$_FAIL, BLANKS_OFF)
;  350          0412  1      );
;  351          0413  1
;  352        P 0414  1  $STATE (NO_COMMENT,
;  353        P 0415  1      ('!', TPA$_FAIL),
;  354        P 0416  1      (TPA$_LAMBDA, TPA$_EXIT)
;  355          0417  1      );
;  356          0418  1
;  357          0419  1  !+
;  358          0420  1  ! A repeated value is of the form n*value, where n is an unsigned integer and
;  359          0421  1  ! no delimiters appear on either side of the "*".  A repeated null is of the
;  360          0422  1  ! form "n*" where a delimiter follows the "*".
;  361          0423  1  !-
;  362          0424  1
;  363        P 0425  1  $STATE (REPEATED_VALUE,
;  364        P 0426  1      (TPA$_DECIMAL, REPEAT2, BLANKS_ON)  ! Value stored in TPA$L_NUMBER
;  365          0427  1      );
;  366          0428  1
;  367        P 0429  1  $STATE (REPEAT2,
;  368        P 0430  1      ('*', REPEAT3, STORE_REPEAT),
;  369        P 0431  1      (TPA$_LAMBDA, TPA$_FAIL, BLANKS_OFF)
;  370          0432  1      );
;  371          0433  1
;  372        P 0434  1  $STATE (REPEAT3,
;  373        P 0435  1      ((VALUE), TPA$_EXIT, END_REPEAT),   ! n*c
;  374        P 0436  1      ((NOT_DELIM), ERROR_STATE),         ! Not n*
;  375        P 0437  1      (TPA$_LAMBDA, TPA$_EXIT, BLANKS_OFF)! Is "n*", skipping will be done by STORE_VALUE
;  376          0438  1      );
;  377          0439  1
;  378          0440  1  !+
;  379          0441  1  ! A value can be one of four types.  Integers look like reals, for our purposes.
;  380          0442  1  ! This state can fail if the current string isn't matched by any of these patterns.
;  381          0443  1  !-
;  382        P 0444  1  $STATE (VALUE,
;  383        P 0445  1      ((LOGICAL), TPA$_EXIT, STORE_LOGICAL),
;  384        P 0446  1      ((REAL), TPA$_EXIT, STORE_REAL),
;  385        P 0447  1      ((COMPLEX), TPA$_EXIT),     ! Stores are done for each part
;  386        P 0448  1      ((CHARACTER), TPA$_EXIT, END_CHARACTER)
;  387          0449  1      );
;  388          0450  1
;  389          0451  1
```

```
;  390          0452  1 !+
;  391          0453  1 ! A variable consists of an identifier, followed by an optional subscript,
;  392          0454  1 ! followed by an optional substring.  If, while parsing values for the previous
;  393          0455  1 ! assignment, it was determined that the last "value" was really an identifier,
;  394          0456  1 ! WAS_VALUE_IDENT will retrieve the token from NML$T_TOKEN and call LOOKUP_IDENTIFIER
;  395          0457  1 ! itself.  Otherwise, we look for an identifier here.
;  396          0458  1 !-
;  397          0459  1
;  398        P 0460  1 $STATE (VARIABLE,
;  399        P 0461  1     (TPA$_LAMBDA, VARIABLE2, WAS_VALUE_IDENT),  ! Fails if last token was not
;  400        P 0462  1                                                 ! an identifier.  If it succeeds,
;  401        P 0463  1                                                 ! lookup is done.
;  402        P 0464  1     ((IDENTIFIER), VARIABLE2, LOOKUP_IDENTIFIER),
;  403          0465  1     );
;  404          0466  1
;  405        P 0467  1 $STATE (VARIABLE2,
;  406        P 0468  1     (TPA$_LAMBDA, , BLANKS_ON)
;  407          0469  1     );
;  408          0470  1
;  409          0471  1 !+
;  410          0472  1 ! Look for subscript or substring.
;  411          0473  1 !-
;  412        P 0474  1 $STATE (SUBSCRIPT_START,
;  413        P 0475  1     ('(', SUB_LOOP1, INIT_SUBS),          ! Signals error if subscript/substring not ok
;  414        P 0476  1     (TPA$_LAMBDA, TPA$_EXIT)
;  415          0477  1     );
;  416          0478  1
;  417          0479  1 !+
;  418          0480  1 ! Get first subscript or first substring.  We can't tell which is which until
;  419          0481  1 ! we see the ":".
;  420          0482  1 !-
;  421        P 0483  1 $STATE (SUB_LOOP1,
;  422        P 0484  1     ((END_OF_LINE), SUB_LOOP1, NEXT_RECORD),
;  423        P 0485  1     (TPA$_BLANK, SUB_LOOP1),
;  424        P 0486  1     ((DECIMAL_INTEGER), , STORE_SUBS),
;  425        P 0487  1     (':', RIGHT_SUBSTRING, SUBSTRING_COLON),    ! Succeeds if substring ok
;  426        P 0488  1                                                 ! otherwise signals FOR$_INVREFVAR
;  427        P 0489  1     (TPA$_LAMBDA, INVREFVAR_STATE)              ! Signal FOR$_INVREFVAR
;  428          0490  1     );
;  429          0491  1
;  430          0492  1 !+
;  431          0493  1 ! This state and the next one consist of the loop looking for subscripts.
;  432          0494  1 ! if a colon is found, control transfers to the substring processor.
;  433          0495  1 !-
;  434        P 0496  1 $STATE (SUB_LOOP2,
;  435        P 0497  1     ((END_OF_LINE), SUB_LOOP2, NEXT_RECORD),
;  436        P 0498  1     (TPA$_BLANK, SUB_LOOP2),
;  437        P 0499  1     (',', SUB_LOOP3),
;  438        P 0500  1     (':', RIGHT_SUBSTRING, SUBSTRING_COLON),    ! Succeeds if substring ok
;  439        P 0501  1                                                 ! otherwise signals FOR$_INVREFVAR
;  440        P 0502  1     (')', START_SUBSTRING, END_SUBSCRIPT),
;  441        P 0503  1     (TPA$_LAMBDA, ERROR_STATE)
;  442          0504  1     );
;  443          0505  1
;  444        P 0506  1 $STATE (SUB_LOOP3,
;  445        P 0507  1     ((END_OF_LINE), SUB_LOOP3, NEXT_RECORD),
;  446        P 0508  1     (TPA$_BLANK, SUB_LOOP3),
```

```
;  447          P 0509  1      ((DECIMAL_INTEGER), SUB_LOOP2, STORE_SUBS),
;  448          P 0510  1      (TPA$_LAMBDA, INVREFVAR_STATE)                  ! Signal FOR$_INVREFVAR
;  449            0511  1      );
;  450            0512  1
;  451            0513  1 !+
;  452            0514  1 ! This state is reached if we have already processed a subscript.  At this point,
;  453            0515  1 ! only a substring is allowed.
;  454            0516  1 !-
;  455          P 0517  1 $STATE (START_SUBSTRING,
;  456          P 0518  1      ('(' , INIT_SUBS),
;  457          P 0519  1      (TPA$_LAMBDA, TPA$_EXIT)
;  458            0520  1      );
;  459            0521  1
;  460          P 0522  1 $STATE (LEFT_SUBSTRING,
;  461          P 0523  1      ((END_OF_LINE), LEFT_SUBSTRING, NEXT_RECORD),
;  462          P 0524  1      (TPA$_BLANK, LEFT_SUBSTRING),
;  463          P 0525  1      ((DECIMAL_INTEGER), SUBSTRING2, STORE_SUBS),
;  464          P 0526  1      (':', RIGHT_SUBSTRING, SUBSTRING_COLON),
;  465          P 0527  1      (TPA$_LAMBDA, INVREFVAR_STATE)                  ! Signal FOR$_INVREFVAR
;  466            0528  1      );
;  467            0529  1
;  468          P 0530  1 $STATE (SUBSTRING2,
;  469          P 0531  1      ((END_OF_LINE), SUBSTRING2, NEXT_RECORD),
;  470          P 0532  1      (TPA$_BLANK, SUBSTRING2),
;  471          P 0533  1      (':', RIGHT_SUBSTRING, SUBSTRING_COLON),
;  472          P 0534  1      (TPA$_LAMBDA, ERROR_STATE)
;  473            0535  1      );
;  474            0536  1
;  475          P 0537  1 $STATE (RIGHT_SUBSTRING,
;  476          P 0538  1      ((END_OF_LINE), RIGHT_SUBSTRING, NEXT_RECORD),
;  477          P 0539  1      (TPA$_BLANK, RIGHT_SUBSTRING),
;  478          P 0540  1      ((DECIMAL_INTEGER), SUBSTRING3, STORE_SUBS),
;  479          P 0541  1      (')', TPA$_EXIT, END_SUBSTRING),
;  480          P 0542  1      (TPA$_LAMBDA, INVREFVAR_STATE)                  ! Signal FOR$_INVREFVAR
;  481            0543  1      );
;  482            0544  1
;  483          P 0545  1 $STATE (SUBSTRING3,
;  484          P 0546  1      ((END_OF_LINE), SUBSTRING3, NEXT_RECORD),
;  485          P 0547  1      (TPA$_BLANK, SUBSTRING3),
;  486          P 0548  1      (')', TPA$_EXIT, END_SUBSTRING),
;  487          P 0549  1      (TPA$_LAMBDA, ERROR_STATE)
;  488            0550  1      );
;  489            0551  1
;  490            0552  1 !+
;  491            0553  1 ! An identifier is a letter followed by 0 or more letters, digits, "$" or "_".
;  492            0554  1 !-
;  493          P 0555  1 $STATE (IDENTIFIER,
;  494          P 0556  1      (TPA$_ALPHA, , BLANKS_ON)
;  495            0557  1      );
;  496            0558  1
;  497          P 0559  1 $STATE (,
;  498          P 0560  1      (TPA$_SYMBOL, TPA$_EXIT, BLANKS_OFF),           ! Matches any string whose characters
;  499          P 0561  1                                                      ! consist of letters, digits,
;  500          P 0562  1                                                      ! "$" and "_".
;  501          P 0563  1      (TPA$_LAMBDA, TPA$_EXIT, BLANKS_OFF),
;  502            0564  1      );
;  503            0565  1
```

```
;  504        P 0566  1 $STATE (DECIMAL_INTEGER,
;  505        P 0567  1     ((INTEGER), TPA$_EXIT, CONVERT_INTEGER)
;  506          0568  1     );
;  507          0569  1
;  508        P 0570  1 $STATE (INTEGER,
;  509        P 0571  1     ('+', , BLANKS_ON),
;  510        P 0572  1     ('-', , BLANKS_ON),
;  511        P 0573  1     (TPA$_LAMBDA, , BLANKS_ON)
;  512          0574  1     );
;  513          0575  1
;  514        P 0576  1 $STATE (,
;  515        P 0577  1     (TPA$_DECIMAL, TPA$_EXIT, BLANKS_OFF),
;  516        P 0578  1     (TPA$_LAMBDA, TPA$_FAIL, BLANKS_OFF)
;  517          0579  1     );
;  518          0580  1
;  519          0581  1 !+
;  520          0582  1 ! Pattern for a REAL value.
;  521          0583  1 !-
;  522        P 0584  1 $STATE (REAL,
;  523        P 0585  1     ('+', , BLANKS_ON),
;  524        P 0586  1     ('-', , BLANKS_ON),
;  525        P 0587  1     (TPA$_LAMBDA, , BLANKS_ON)
;  526          0588  1     );
;  527          0589  1
;  528        P 0590  1 $STATE (REAL1,
;  529        P 0591  1     (TPA$_DIGIT, REAL1),
;  530        P 0592  1     ('.'),
;  531        P 0593  1     (TPA$_LAMBDA)
;  532          0594  1     );
;  533          0595  1
;  534        P 0596  1 $STATE (REAL2,
;  535        P 0597  1     (TPA$_DIGIT, REAL2),
;  536        P 0598  1     (TPA$_LAMBDA)
;  537          0599  1     );
;  538          0600  1
;  539        P 0601  1 $STATE (EXPONENT,
;  540        P 0602  1     ('E'),
;  541        P 0603  1     ('e'),
;  542        P 0604  1     ('D'),
;  543        P 0605  1     ('d'),
;  544        P 0606  1     ('Q'),
;  545        P 0607  1     ('q'),
;  546        P 0608  1     (TPA$_LAMBDA)
;  547          0609  1     );
;  548          0610  1
;  549        P 0611  1 $STATE (,
;  550        P 0612  1     ('+'),
;  551        P 0613  1     ('-'),
;  552        P 0614  1     (TPA$_LAMBDA)
;  553          0615  1     );
;  554          0616  1
;  555        P 0617  1 $STATE (EXPONENT2,
;  556        P 0618  1     (TPA$_DIGIT, EXPONENT2),
;  557        P 0619  1     (TPA$_LAMBDA)
;  558          0620  1     );
;  559          0621  1
;  560        P 0622  1 $STATE (,                         ! Fail if next character is not a delimiter
```

```
;  561        P 0623  1      ((NOT_DELIM), TPA$_FAIL),   ! but don't consume the character.
;  562        P 0624  1      (TPA$_LAMBDA, TPA$_EXIT)
;  563          0625  1      );
;  564          0626  1
;  565        P 0627  1  $STATE (NOT_DELIM,                 ! Fails if next character is a delimiter
;  566        P 0628  1      ((END_OF_LINE), TPA$_FAIL),
;  567        P 0629  1      (TPA$_BLANK, TPA$_FAIL),
;  568        P 0630  1      (' ', TPA$_FAIL),
;  569        P 0631  1      ('$', TPA$_FAIL),
;  570        P 0632  1      ('&', TPA$_FAIL),
;  571        P 0633  1      (')', TPA$_FAIL),           ! Can show in complex values
;  572        P 0634  1      (TPA$_LAMBDA, TPA$_EXIT)
;  573          0635  1      );
;  574          0636  1
;  575          0637  1  !+
;  576          0638  1  ! Pattern for a logical value.  It is complex because any string can follow
;  577          0639  1  ! after the initial T, F, .T or .F up to the next "delimiter".
;  578          0640  1  !-
;  579        P 0641  1  $STATE (LOGICAL,
;  580        P 0642  1      ('.', , BLANKS_ON),
;  581        P 0643  1      (TPA$_LAMBDA, , BLANKS_ON)
;  582          0644  1      );
;  583          0645  1
;  584        P 0646  1  $STATE (,
;  585        P 0647  1      ('T'),
;  586        P 0648  1      ('t'),
;  587        P 0649  1      ('F'),
;  588        P 0650  1      ('f')
;  589          0651  1      );
;  590          0652  1
;  591          0653  1  !+
;  592          0654  1  ! Consume characters up to but not including the next delimiter.
;  593          0655  1  !-
;  594        P 0656  1  $STATE (LOGICAL1,
;  595        P 0657  1      ((LOGICAL2), LOGICAL1),
;  596        P 0658  1      (TPA$_LAMBDA, TPA$_EXIT, BLANKS_OFF)
;  597          0659  1      );
;  598          0660  1
;  599          0661  1  !+
;  600          0662  1  ! Indicates by failing if any of the selected characters are found.
;  601          0663  1  !-
;  602        P 0664  1  $STATE (LOGICAL2,
;  603        P 0665  1      ((END_OF_LINE), TPA$_FAIL),
;  604        P 0666  1      (TPA$_BLANK, TPA$_FAIL),
;  605        P 0667  1      (' ', TPA$_FAIL),
;  606        P 0668  1      ('(', TPA$_FAIL),
;  607        P 0669  1      ('=', TPA$_FAIL),
;  608        P 0670  1      ('$', TPA$_FAIL),
;  609        P 0671  1      ('&', TPA$_FAIL),
;  610        P 0672  1      (TPA$_ANY, TPA$_EXIT)
;  611          0673  1      );
;  612          0674  1
;  613          0675  1  !+
;  614          0676  1  ! Parse and store the representation of a complex value.  This is safe because
;  615          0677  1  ! a complex value can not possibly be an identifier.
;  616        P 0678  1  $STATE (COMPLEX,
;  617        P 0679  1      ('(', COMPLEX2)
```

```
;  618          0680  1      );
;  619          0681  1
;  620        P 0682  1  $STATE (COMPLEX2,
;  621        P 0683  1      ((END_OF_LINE), COMPLEX2, NEXT_RECORD),
;  622        P 0684  1      (TPA$_BLANK, COMPLEX2),
;  623        P 0685  1      ((REAL), COMPLEX3, STORE_COMPLEX),  ! Store real part
;  624        P 0686  1      (TPA$_LAMBDA, ERROR_STATE)
;  625          0687  1      );
;  626          0688  1
;  627        P 0689  1  $STATE (COMPLEX3,
;  628        P 0690  1      ((END_OF_LINE), COMPLEX3, NEXT_RECORD),
;  629        P 0691  1      (TPA$_BLANK, COMPLEX3),
;  630        P 0692  1      (',', COMPLEX4),
;  631        P 0693  1      (TPA$_LAMBDA, ERROR_STATE)
;  632          0694  1      );
;  633          0695  1
;  634        P 0696  1  $STATE (COMPLEX4,
;  635        P 0697  1      ((END_OF_LINE), COMPLEX4, NEXT_RECORD),
;  636        P 0698  1      (TPA$_BLANK, COMPLEX4),
;  637        P 0699  1      ((REAL), COMPLEX5, STORE_COMPLEX),  ! Store imaginary part
;  638        P 0700  1      (TPA$_LAMBDA, ERROR_STATE)
;  639          0701  1      );
;  640          0702  1
;  641        P 0703  1  $STATE (COMPLEX5,
;  642        P 0704  1      ((END_OF_LINE), COMPLEX5, NEXT_RECORD),
;  643        P 0705  1      (TPA$_BLANK, COMPLEX5),
;  644        P 0706  1      (')', TPA$_EXIT),
;  645        P 0707  1      (TPA$_LAMBDA, ERROR_STATE)
;  646          0708  1      );
;  647          0709  1
;  648          0710  1  !+
;  649          0711  1  ! Pattern for a character string.  Inside the string, two consecutive quotes
;  650          0712  1  ! are counted as one.  This value is stored in the user variable as it goes,
;  651          0713  1  ! since this can not possibly be an identifier.
;  652          0714  1  !-
;  653        P 0715  1  $STATE (CHARACTER,
;  654        P 0716  1      (SINGLE_QUOTE, CHARACTER1, STRING_OK)        ! Signals error if not type CHARACTER
;  655          0717  1      );                                           ! Also turns on TPA$V_BLANKS
;  656          0718  1
;  657        P 0719  1  $STATE (CHARACTER1,
;  658        P 0720  1      (TPA$_EOS, CHARACTER1, NEXT_RECORD),         ! Don't use END_OF_LINE because
;  659        P 0721  1      (SINGLE_QUOTE, NEXT_QUOTE),                  ! a "!" is a valid character.
;  660        P 0722  1      (TPA$_ANY, CHARACTER1, STORE_CHARACTER)
;  661          0723  1      );
;  662          0724  1
;  663        P 0725  1  $STATE (NEXT_QUOTE,
;  664        P 0726  1      (TPA$_EOS, NEXT_QUOTE, NEXT_RECORD),         ! Don't use END_OF_LINE.
;  665        P 0727  1      (SINGLE_QUOTE, CHARACTER1, STORE_CHARACTER),
;  666        P 0728  1      (TPA$_LAMBDA, TPA$_EXIT)
;  667          0729  1      );
;  668          0730  1
;  669          0731  1  !+
;  670          0732  1  ! This state is transferred to if a syntax error is detected in the parsing. It
;  671          0733  1  ! calls SYNTAX_ERROR with a token which is at or near where the error was.
;  672          0734  1  ! SYNTAX_ERROR signals FOR$_SYNERRNAM.
;  673          0735  1  !-
;  674        P 0736  1  $STATE (ERROR_STATE,
```

```
;  675          P 0737  1      (TPA$_ANY, TPA$_FAIL, SYNTAX_ERROR),
;  676          P 0738  1      (TPA$_LAMBDA, TPA$_FAIL, SYNTAX_ERROR)
;  677            0739  1      );
;  678            0740  1
;  679            0741  1 !+
;  680            0742  1 ! This state is transferred to when there is some invalid reference on a
;  681            0743  1 ! variable, i.e. subscripting a scalar, substringing a non-character or using
;  682            0744  1 ! non-integers in subscripts/substrings.  It calls INVREFVAR_ERROR which
;  683            0745  1 ! signals FOR$_INVREFVAR.
;  684            0746  1 !-
;  685          P 0747  1 $STATE (INVREFVAR_STATE,
;  686          P 0748  1      (TPA$_LAMBDA, TPA$_FAIL, INVREFVAR_ERROR)
;  687            0749  1      );
;  688            0750  1 !<BLF/PAGE>
```

```
;  690          0751  1 %SBTTL 'NEXT_RECORD - Get next record'
;  691          0752  1 ROUTINE NEXT_RECORD =
;  692          0753  1
;  693          0754  1 !++
;  694          0755  1 ! FUNCTIONAL DESCRIPTION:
;  695          0756  1 !
;  696          0757  1 !       Reads a new record from the current unit and updates the STRING pointers
;  697          0758  1 !       in PARAM_BLOCK.
;  698          0759  1 !
;  699          0760  1 ! CALLING SEQUENCE:
;  700          0761  1 !
;  701          0762  1 !       status = NEXT_RECORD ()
;  702          0763  1 !
;  703          0764  1 ! FORMAL PARAMETERS:
;  704          0765  1 !
;  705          0766  1 !       NONE
;  706          0767  1 !
;  707          0768  1 ! IMPLICIT INPUTS:
;  708          0769  1 !
;  709          0770  1 !       AP      Points to PARAM_BLOCK
;  710          0771  1 !
;  711          0772  1 ! IMPLICIT OUTPUTS:
;  712          0773  1 !
;  713          0774  1 !       PARAM_BLOCK [TPA$L_STRINGPTR] is address of new record
;  714          0775  1 !       PARAM_BLOCK [TPA$L_STRINGCNT] is record length
;  715          0776  1 !
;  716          0777  1 ! COMPLETION STATUS:
;  717          0778  1 !
;  718          0779  1 !       1 for success; all errors are signalled.
;  719          0780  1 !
;  720          0781  1 ! SIDE EFFECTS:
;  721          0782  1 !
;  722          0783  1 !
;  723          0784  1 !
;  724          0785  1 !--
;  725          0786  1
;  726          0787  2     BEGIN
;  727          0788  2
;  728          0789  2     BUILTIN
;  729          0790  2         AP;                 ! Argument pointer points to parameter block
;  730          0791  2
;  731          0792  2     MAP
;  732          0793  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  733          0794  2
;  734          0795  2     GLOBAL REGISTER
;  735          0796  2         CCB = 11: REF $FOR$CCB_DECL;
;  736          0797  2
;  737          0798  2     CCB = .AP [NML$A_CCB];      ! Fetch CCB address
;  738          0799  2     DO
;  739          0800  3         BEGIN
;  740          0801  3         FOR$$REC_RSNO ();                   ! Read the next record
;  741          0802  3         CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + 1; ! Start with second byte
;  742          0803  3         AP [TPA$L_STRINGPTR] = .CCB [LUB$A_BUF_PTR];
;  743          0804  3         AP [TPA$L_STRINGCNT] = .CCB [LUB$A_BUF_END] - .CCB [LUB$A_BUF_PTR];
;  744          0805  3         END
;  745          0806  2     UNTIL .AP [TPA$L_STRINGCNT] GTR 0;
;  746          0807  2     RETURN 1;
```

```
;  747          0808  2
;  748          0809  1     END;


                                                .TITLE  FOR$$NML_TABLES FOR$$NML_TABLES - TPARSE state
;                                                                       tables for NAMEL
                                                .IDENT  \1-012\

                                                .PSECT  _LIB$STATE$,NOWRT,  SHR,  PIC,1

                                        00000 FOR$$A_NMLSTATE::
                                                .BLKB   0
                                        00000 BEGIN_SCAN:
                                                .BLKB   0
                                    99F8  00000 ;TPA$TYPE
                                                U.2:    .WORD   -26120                                  ;
                                    0000* 00002 ;TPA$SUBEXP
                                                U.4:    .WORD   <<U.3-U.4>-2>                           ;
                                00000000* 00004 ;TPA$ACTION
                                                U.5:    .LONG   <<NEXT_RECORD-U.5>-4>                   ;
                                    0000* 00008 ;TPA$TARGET
                                                U.6:    .WORD   <<BEGIN_SCAN-U.6>-2>                    ;
                                    1024  0000A ;TPA$TYPE
                                                U.7:    .WORD   4132                                    ;
                                    FFFF  0000C ;TPA$TARGET
                                                U.8:    .WORD   -1                                      ;
                                    1026  0000E ;TPA$TYPE
                                                U.9:    .WORD   4134                                    ;
                                    FFFF  00010 ;TPA$TARGET
                                                U.10:   .WORD   -1                                      ;
                                    99F8  00012 ;TPA$TYPE
                                                U.11:   .WORD   -26120                                  ;
                                    0000* 00014 ;TPA$SUBEXP
                                                U.13:   .WORD   <<U.12-U.13>-2>                         ;
                                00000000V 00016 ;TPA$ACTION
                                                U.14:   .LONG   <<BLANKS_OFF-U.14>-4>                   ;
                                    0000* 0001A ;TPA$TARGET
                                                U.15:   .WORD   <<BEGIN_SCAN-U.15>-2>                   ;
                                    15F6  0001C ;TPA$TYPE
                                                U.16:   .WORD   5622                                    ;
                                    0000* 0001E ;TPA$TARGET
                                                U.18:   .WORD   <<U.17-U.18>-2>                         ;
                                          00020 ;END_OF_LINE
                                                U.3:    .BLKB   0
                                    11F7  00020 ;TPA$TYPE
                                                U.19:   .WORD   4599                                    ;
                                    FFFF  00022 ;TPA$TARGET
                                                U.20:   .WORD   -1                                      ;
                                    19F8  00024 ;TPA$TYPE
                                                U.21:   .WORD   6648                                    ;
                                    0000* 00026 ;TPA$SUBEXP
                                                U.23:   .WORD   <<U.22-U.23>-2>                         ;
                                    FFFE  00028 ;TPA$TARGET
                                                U.24:   .WORD   -2                                      ;
                                    15F6  0002A ;TPA$TYPE
                                                U.25:   .WORD   5622                                    ;
                                    FFFF  0002C ;TPA$TARGET
```

```
                              U.26:    .WORD   -1                           ;
                   0002E ;ASSIGNMENT
                              U.12:    .BLKB   0
            99F8   0002E ;TPA$TYPE
                              U.27:    .WORD   -26120                       ;
            0000*  00030 ;TPA$SUBEXP
                              U.29:    .WORD   <<U.28-U.29>-2>              ;
        00000000V  00032 ;TPA$ACTION
                              U.30:    .LONG   <<BLANKS_OFF-U.30>-4>        ;
            0000*  00036 ;TPA$TARGET
                              U.32:    .WORD   <<U.31-U.32>-2>              ;
            903F   00038 ;TPA$TYPE
                              U.33:    .WORD   -28609                       ;
        00000000V  0003A ;TPA$ACTION
                              U.34:    .LONG   <<DUMP_NAMES-U.34>-4>        ;
            0000*  0003E ;TPA$TARGET
                              U.36:    .WORD   <<U.35-U.36>-2>              ;
            99F8   00040 ;TPA$TYPE
                              U.37:    .WORD   -26120                       ;
            0000*  00042 ;TPA$SUBEXP
                              U.39:    .WORD   <<U.38-U.39>-2>              ;
        00000000V  00044 ;TPA$ACTION
                              U.40:    .LONG   <<DUMP_VALUES-U.40>-4>       ;
            0000*  00048 ;TPA$TARGET
                              U.41:    .WORD   <<U.35-U.41>-2>              ;
            15F6   0004A ;TPA$TYPE
                              U.42:    .WORD   5622                         ;
            0000*  0004C ;TPA$TARGET
                              U.43:    .WORD   <<U.17-U.43>-2>              ;
                   0004E ;FLUSH_RECORD
                              U.35:    .BLKB   0
            11F7   0004E ;TPA$TYPE
                              U.44:    .WORD   4599                         ;
            FFFF   00050 ;TPA$TARGET
                              U.45:    .WORD   -1                           ;
            15ED   00052 ;TPA$TYPE
                              U.46:    .WORD   5613                         ;
            0000*  00054 ;TPA$TARGET
                              U.47:    .WORD   <<U.35-U.47>-2>              ;
                   00056 ;ASSN_EQL
                              U.31:    .BLKB   0
            99F8   00056 ;TPA$TYPE
                              U.48:    .WORD   -26120                       ;
            0000*  00058 ;TPA$SUBEXP
                              U.49:    .WORD   <<U.3-U.49>-2>               ;
        00000000*  0005A ;TPA$ACTION
                              U.50:    .LONG   <<NEXT_RECORD-U.50>-4>       ;
            0000*  0005E ;TPA$TARGET
                              U.51:    .WORD   <<U.31-U.51>-2>              ;
            103D   00060 ;TPA$TYPE
                              U.52:    .WORD   4157                         ;
            0000*  00062 ;TPA$TARGET
                              U.54:    .WORD   <<U.53-U.54>-2>              ;
            15F6   00064 ;TPA$TYPE
                              U.55:    .WORD   5622                         ;
            0000*  00066 ;TPA$TARGET
                              U.56:    .WORD   <<U.17-U.56>-2>              ;
```

```
                    00068 ;VALUE_LIST
                          U.53:    .BLKB   0
                99F8  00068 ;TPA$TYPE
                          U.57:    .WORD   -26120                                  ;
                0000* 0006A ;TPA$SUBEXP
                          U.58:    .WORD   <<U.3-U.58>-2>                          ;
            00000000* 0006C ;TPA$ACTION
                          U.59:    .LONG   <<NEXT_RECORD-U.59>-4>                  ;
                0000* 00070 ;TPA$TARGET
                          U.60:    .WORD   <<U.53-U.60>-2>                         ;
                902C  00072 ;TPA$TYPE
                          U.61:    .WORD   -28628                                  ;
            00000000V 00074 ;TPA$ACTION
                          U.62:    .LONG   <<NULL_VALUE-U.62>-4>                   ;
                0000* 00078 ;TPA$TARGET
                          U.63:    .WORD   <<U.53-U.63>-2>                         ;
                99F8  0007A ;TPA$TYPE
                          U.64:    .WORD   -26120                                  ;
                0000* 0007C ;TPA$SUBEXP
                          U.66:    .WORD   <<U.65-U.66>-2>                         ;
            00000000V 0007E ;TPA$ACTION
                          U.67:    .LONG   <<BLANKS_ON-U.67>-4>                    ;
                0000* 00082 ;TPA$TARGET
                          U.69:    .WORD   <<U.68-U.69>-2>                         ;
                99F8  00084 ;TPA$TYPE
                          U.70:    .WORD   -26120                                  ;
                0000* 00086 ;TPA$SUBEXP
                          U.72:    .WORD   <<U.71-U.72>-2>                         ;
            00000000V 00088 ;TPA$ACTION
                          U.73:    .LONG   <<BLANKS_ON-U.73>-4>                    ;
                0000* 0008C ;TPA$TARGET
                          U.74:    .WORD   <<U.68-U.74>-2>                         ;
                15F6  0008E ;TPA$TYPE
                          U.75:    .WORD   5622                                    ;
                FFFF  00090 ;TPA$TARGET
                          U.76:    .WORD   -1                                      ;
                    00092 ;VALUE_LIST1
                          U.68:    .BLKB   0
                99F8  00092 ;TPA$TYPE
                          U.77:    .WORD   -26120                                  ;
                0000* 00094 ;TPA$SUBEXP
                          U.78:    .WORD   <<U.3-U.78>-2>                          ;
            00000000V 00096 ;TPA$ACTION
                          U.79:    .LONG   <<BLANKS_OFF-U.79>-4>                   ;
                0000* 0009A ;TPA$TARGET
                          U.81:    .WORD   <<U.80-U.81>-2>                         ;
                91F2  0009C ;TPA$TYPE
                          U.82:    .WORD   -28174                                  ;
            00000000V 0009E ;TPA$ACTION
                          U.83:    .LONG   <<BLANKS_OFF-U.83>-4>                   ;
                0000* 000A2 ;TPA$TARGET
                          U.84:    .WORD   <<U.80-U.84>-2>                         ;
                99F8  000A4 ;TPA$TYPE
                          U.85:    .WORD   -26120                                  ;
                0000* 000A6 ;TPA$SUBEXP
                          U.87:    .WORD   <<U.86-U.87>-2>                         ;
            00000000V 000A8 ;TPA$ACTION
```

```
                                                    U.88:   .LONG   <<BLANKS_OFF-U.88>-4>                   ;
                                          0000*  000AC ;TPA$TARGET
                                                    U.89:   .WORD   <<U.80-U.89>-2>                         ;
                                           95F6  000AE ;TPA$TYPE
                                                    U.90:   .WORD   -27146                                  ;
                                      00000000V  000B0 ;TPA$ACTION
                                                    U.91:   .LONG   <<SET_VALUE_IDENT-U.91>-4>              ;
                                           FFFF  000B4 ;TPA$TARGET
                                                    U.92:   .WORD   -1                                      ;
                                                 000B6 ;VALUE_LIST2
                                                    U.80:   .BLKB   0
                                           99F8  000B6 ;TPA$TYPE
                                                    U.93:   .WORD   -26120                                  ;
                                          0000*  000B8 ;TPA$SUBEXP
                                                    U.94:   .WORD   <<U.3-U.94>-2>                          ;
                                      00000000*  000BA ;TPA$ACTION
                                                    U.95:   .LONG   <<NEXT_RECORD-U.95>-4>                  ;
                                          0000*  000BE ;TPA$TARGET
                                                    U.96:   .WORD   <<U.80-U.96>-2>                         ;
                                           11F2  000C0 ;TPA$TYPE
                                                    U.97:   .WORD   4594                                    ;
                                          0000*  000C2 ;TPA$TARGET
                                                    U.98:   .WORD   <<U.80-U.98>-2>                         ;
                                           902C  000C4 ;TPA$TYPE
                                                    U.99:   .WORD   -28628                                  ;
                                      00000000V  000C6 ;TPA$ACTION
                                                    U.100:  .LONG   <<STORE_VALUE-U.100>-4>                 ;
                                          0000*  000CA ;TPA$TARGET
                                                    U.101:  .WORD   <<U.53-U.101>-2>                        ;
                                           99F8  000CC ;TPA$TYPE
                                                    U.102:  .WORD   -26120                                  ;
                                          0000*  000CE ;TPA$SUBEXP
                                                    U.104:  .WORD   <<U.103-U.104>-2>                       ;
                                      00000000V  000D0 ;TPA$ACTION
                                                    U.105:  .LONG   <<STORE_VALUE-U.105>-4>                 ;
                                          0000*  000D4 ;TPA$TARGET
                                                    U.106:  .WORD   <<U.53-U.106>-2>                        ;
                                           95F6  000D6 ;TPA$TYPE
                                                    U.107:  .WORD   -27146                                  ;
                                      00000000V  000D8 ;TPA$ACTION
                                                    U.108:  .LONG   <<SET_VALUE_IDENT-U.108>-4>             ;
                                           FFFF  000DC ;TPA$TARGET
                                                    U.109:  .WORD   -1                                      ;
                                                 000DE ;NO_LPAREN
                                                    U.86:   .BLKB   0
                                           1028  000DE ;TPA$TYPE
                                                    U.110:  .WORD   4136                                    ;
                                           FFFE  000E0 ;TPA$TARGET
                                                    U.111:  .WORD   -2                                      ;
                                           15F6  000E2 ;TPA$TYPE
                                                    U.112:  .WORD   5622                                    ;
                                           FFFF  000E4 ;TPA$TARGET
                                                    U.113:  .WORD   -1                                      ;
                                                 000E6 ;NO_EQUALS
                                                    U.103:  .BLKB   0
                                           19F8  000E6 ;TPA$TYPE
                                                    U.114:  .WORD   6648                                    ;
```

```
                                    0000* 000E8 ;TPA$SUBEXP
                                                U.116:   .WORD   <<U.115-U.116>-2>                                ;
                                    0000* 000EA ;TPA$TARGET
                                                U.118:   .WORD   <<U.117-U.118>-2>                                ;
                                    15F6  000EC ;TPA$TYPE
                                                U.119:   .WORD   5622                                             ;
                                    FFFF  000EE ;TPA$TARGET
                                                U.120:   .WORD   -1                                               ;
                                          000F0 ;NO_EQUALS2
                                                U.117:   .BLKB   0
                                    103D  000F0 ;TPA$TYPE
                                                U.121:   .WORD   4157                                             ;
                                    FFFE  000F2 ;TPA$TARGET
                                                U.122:   .WORD   -2                                               ;
                                    15F6  000F4 ;TPA$TYPE
                                                U.123:   .WORD   5622                                             ;
                                    FFFF  000F6 ;TPA$TARGET
                                                U.124:   .WORD   -1                                               ;
                                          000F8 ;NO_EQUALS_QUESTION
                                                U.115:   .BLKB   0
                                    19F8  000F8 ;TPA$TYPE
                                                U.125:   .WORD   6648                                             ;
                                    0000* 000FA ;TPA$SUBEXP
                                                U.126:   .WORD   <<U.38-U.126>-2>                                 ;
                                    FFFE  000FC ;TPA$TARGET
                                                U.127:   .WORD   -2                                               ;
                                    15F6  000FE ;TPA$TYPE
                                                U.128:   .WORD   5622                                             ;
                                    FFFF  00100 ;TPA$TARGET
                                                U.129:   .WORD   -1                                               ;
                                          00102 ;EQUALS_QUESTION
                                                U.38:    .BLKB   0
                                    843D  00102 ;TPA$TYPE
                                                U.130:   .WORD   -31683                                           ;
                                00000000V 00104 ;TPA$ACTION
                                                U.131:   .LONG   <<BLANKS_ON-U.131>-4>                            ;
                                    903F  00108 ;TPA$TYPE
                                                U.132:   .WORD   -28609                                           ;
                                00000000V 0010A ;TPA$ACTION
                                                U.133:   .LONG   <<BLANKS_OFF-U.133>-4>                           ;
                                    FFFF  0010E ;TPA$TARGET
                                                U.134:   .WORD   -1                                               ;
                                    95F6  00110 ;TPA$TYPE
                                                U.135:   .WORD   -27146                                           ;
                                00000000V 00112 ;TPA$ACTION
                                                U.136:   .LONG   <<BLANKS_OFF-U.136>-4>                           ;
                                    FFFE  00116 ;TPA$TARGET
                                                U.137:   .WORD   -2                                               ;
                                          00118 ;NO_COMMENT
                                                U.22:    .BLKB   0
                                    1021  00118 ;TPA$TYPE
                                                U.138:   .WORD   4129                                             ;
                                    FFFE  0011A ;TPA$TARGET
                                                U.139:   .WORD   -2                                               ;
                                    15F6  0011C ;TPA$TYPE
                                                U.140:   .WORD   5622                                             ;
                                    FFFF  0011E ;TPA$TARGET
```

```
                    U.141:  .WORD   -1                              ;
          00120 ;REPEATED_VALUE
                    U.65:   .BLKB   0
     95F3 00120 ;TPA$TYPE
                    U.142:  .WORD   -27149                          ;
00000000V 00122 ;TPA$ACTION
                    U.143:  .LONG   <<BLANKS_ON-U.143>-4>           ;
    0000* 00126 ;TPA$TARGET
                    U.145:  .WORD   <<U.144-U.145>-2>               ;
          00128 ;REPEAT2
                    U.144:  .BLKB   0
     902A 00128 ;TPA$TYPE
                    U.146:  .WORD   -28630                          ;
00000000V 0012A ;TPA$ACTION
                    U.147:  .LONG   <<STORE_REPEAT-U.147>-4>        ;
    0000* 0012E ;TPA$TARGET
                    U.149:  .WORD   <<U.148-U.149>-2>               ;
     95F6 00130 ;TPA$TYPE
                    U.150:  .WORD   -27146                          ;
00000000V 00132 ;TPA$ACTION
                    U.151:  .LONG   <<BLANKS_OFF-U.151>-4>          ;
     FFFE 00136 ;TPA$TARGET
                    U.152:  .WORD   -2                              ;
          00138 ;REPEAT3
                    U.148:  .BLKB   0
     99F8 00138 ;TPA$TYPE
                    U.153:  .WORD   -26120                          ;
    0000* 0013A ;TPA$SUBEXP
                    U.154:  .WORD   <<U.71-U.154>-2>                ;
00000000V 0013C ;TPA$ACTION
                    U.155:  .LONG   <<END_REPEAT-U.155>-4>          ;
     FFFF 00140 ;TPA$TARGET
                    U.156:  .WORD   -1                              ;
     19F8 00142 ;TPA$TYPE
                    U.157:  .WORD   6648                            ;
    0000* 00144 ;TPA$SUBEXP
                    U.159:  .WORD   <<U.158-U.159>-2>               ;
    0000* 00146 ;TPA$TARGET
                    U.160:  .WORD   <<U.17-U.160>-2>                ;
     95F6 00148 ;TPA$TYPE
                    U.161:  .WORD   -27146                          ;
00000000V 0014A ;TPA$ACTION
                    U.162:  .LONG   <<BLANKS_OFF-U.162>-4>          ;
     FFFF 0014E ;TPA$TARGET
                    U.163:  .WORD   -1                              ;
          00150 ;VALUE
                    U.71:   .BLKB   0
     99F8 00150 ;TPA$TYPE
                    U.164:  .WORD   -26120                          ;
    0000* 00152 ;TPA$SUBEXP
                    U.166:  .WORD   <<U.165-U.166>-2>               ;
00000000V 00154 ;TPA$ACTION
                    U.167:  .LONG   <<STORE_LOGICAL-U.167>-4>       ;
     FFFF 00158 ;TPA$TARGET
                    U.168:  .WORD   -1                              ;
     99F8 0015A ;TPA$TYPE
                    U.169:  .WORD   -26120                          ;
```

```
        0000* 0015C ;TPA$SUBEXP
                    U.171:  .WORD   <<U.170-U.171>-2>                   ;
    00000000V 0015E ;TPA$ACTION
                    U.172:  .LONG   <<STORE_REAL-U.172>-4>              ;
         FFFF 00162 ;TPA$TARGET
                    U.173:  .WORD   -1                                  ;
         19F8 00164 ;TPA$TYPE
                    U.174:  .WORD   6648                                ;
        0000* 00166 ;TPA$SUBEXP
                    U.176:  .WORD   <<U.175-U.176>-2>                   ;
         FFFF 00168 ;TPA$TARGET
                    U.177:  .WORD   -1                                  ;
         9DF8 0016A ;TPA$TYPE
                    U.178:  .WORD   -25096                              ;
        0000* 0016C ;TPA$SUBEXP
                    U.180:  .WORD   <<U.179-U.180>-2>                   ;
    00000000V 0016E ;TPA$ACTION
                    U.181:  .LONG   <<END_CHARACTER-U.181>-4>           ;
         FFFF 00172 ;TPA$TARGET
                    U.182:  .WORD   -1                                  ;
              00174 ;VARIABLE
                    U.28:   .BLKB   0
         91F6 00174 ;TPA$TYPE
                    U.183:  .WORD   -28170                              ;
    00000000V 00176 ;TPA$ACTION
                    U.184:  .LONG   <<WAS_VALUE_IDENT-U.184>-4>         ;
        0000* 0017A ;TPA$TARGET
                    U.186:  .WORD   <<U.185-U.186>-2>                   ;
         9DF8 0017C ;TPA$TYPE
                    U.187:  .WORD   -25096                              ;
        0000* 0017E ;TPA$SUBEXP
                    U.189:  .WORD   <<U.188-U.189>-2>                   ;
    00000000V 00180 ;TPA$ACTION
                    U.190:  .LONG   <<LOOKUP_IDENTIFIER-U.190>-4>       ;
        0000* 00184 ;TPA$TARGET
                    U.191:  .WORD   <<U.185-U.191>-2>                   ;
              00186 ;VARIABLE2
                    U.185:  .BLKB   0
         85F6 00186 ;TPA$TYPE
                    U.192:  .WORD   -31242                              ;
    00000000V 00188 ;TPA$ACTION
                    U.193:  .LONG   <<BLANKS_ON-U.193>-4>               ;
              0018C SUBSCRIPT_START:
                            .BLKB   0
         9028 0018C ;TPA$TYPE
                    U.194:  .WORD   -28632                              ;
    00000000V 0018E ;TPA$ACTION
                    U.195:  .LONG   <<INIT_SUBS-U.195>-4>               ;
        0000* 00192 ;TPA$TARGET
                    U.197:  .WORD   <<U.196-U.197>-2>                   ;
         15F6 00194 ;TPA$TYPE
                    U.198:  .WORD   5622                                ;
         FFFF 00196 ;TPA$TARGET
                    U.199:  .WORD   -1                                  ;
              00198 ;SUB_LOOP1
                    U.196:  .BLKB   0
         99F8 00198 ;TPA$TYPE
```

```
                                  U.200:    .WORD    -26120                                    ;
          0000*  0019A  ;TPA$SUBEXP
                                  U.201:    .WORD    <<U.3-U.201>-2>                           ;
      00000000*  0019C  ;TPA$ACTION
                                  U.202:    .LONG    <<NEXT_RECORD-U.202>-4>                   ;
          0000*  001A0  ;TPA$TARGET
                                  U.203:    .WORD    <<U.196-U.203>-2>                         ;
           11F2  001A2  ;TPA$TYPE
                                  U.204:    .WORD    4594                                      ;
          0000*  001A4  ;TPA$TARGET
                                  U.205:    .WORD    <<U.196-U.205>-2>                         ;
           89F8  001A6  ;TPA$TYPE
                                  U.206:    .WORD    -30216                                    ;
          0000*  001A8  ;TPA$SUBEXP
                                  U.208:    .WORD    <<U.207-U.208>-2>                         ;
      00000000V  001AA  ;TPA$ACTION
                                  U.209:    .LONG    <<STORE_SUBS-U.209>-4>                    ;
           903A  001AE  ;TPA$TYPE
                                  U.210:    .WORD    -28614                                    ;
      00000000V  001B0  ;TPA$ACTION
                                  U.211:    .LONG    <<SUBSTRING_COLON-U.211>-4>               ;
          0000*  001B4  ;TPA$TARGET
                                  U.213:    .WORD    <<U.212-U.213>-2>                         ;
           15F6  001B6  ;TPA$TYPE
                                  U.214:    .WORD    5622                                      ;
          0000*  001B8  ;TPA$TARGET
                                  U.216:    .WORD    <<U.215-U.216>-2>                         ;
                 001BA  SUB_LOOP2:
                                            .BLKB    0
           99F8  001BA  ;TPA$TYPE
                                  U.217:    .WORD    -26120                                    ;
          0000*  001BC  ;TPA$SUBEXP
                                  U.218:    .WORD    <<U.3-U.218>-2>                           ;
      00000000*  001BE  ;TPA$ACTION
                                  U.219:    .LONG    <<NEXT_RECORD-U.219>-4>                   ;
          0000*  001C2  ;TPA$TARGET
                                  U.220:    .WORD    <<SUB_LOOP2-U.220>-2>                     ;
           11F2  001C4  ;TPA$TYPE
                                  U.221:    .WORD    4594                                      ;
          0000*  001C6  ;TPA$TARGET
                                  U.222:    .WORD    <<SUB_LOOP2-U.222>-2>                     ;
           102C  001C8  ;TPA$TYPE
                                  U.223:    .WORD    4140                                      ;
          0000*  001CA  ;TPA$TARGET
                                  U.225:    .WORD    <<U.224-U.225>-2>                         ;
           903A  001CC  ;TPA$TYPE
                                  U.226:    .WORD    -28614                                    ;
      00000000V  001CE  ;TPA$ACTION
                                  U.227:    .LONG    <<SUBSTRING_COLON-U.227>-4>               ;
          0000*  001D2  ;TPA$TARGET
                                  U.228:    .WORD    <<U.212-U.228>-2>                         ;
           9029  001D4  ;TPA$TYPE
                                  U.229:    .WORD    -28631                                    ;
      00000000V  001D6  ;TPA$ACTION
                                  U.230:    .LONG    <<END_SUBSCRIPT-U.230>-4>                 ;
          0000*  001DA  ;TPA$TARGET
                                  U.232:    .WORD    <<U.231-U.232>-2>                         ;
```

```
                                    15F6  001DC ;TPA$TYPE
                                                U.233:  .WORD   5622                                    ;
                                    0000* 001DE ;TPA$TARGET
                                                U.234:  .WORD   <<U.17-U.234>-2>                        ;
                                          001E0 ;SUB_LOOP3
                                                U.224:  .BLKB   0
                                    99F8  001E0 ;TPA$TYPE
                                                U.235:  .WORD   -26120                                  ;
                                    0000* 001E2 ;TPA$SUBEXP
                                                U.236:  .WORD   <<U.3-U.236>-2>                         ;
                                00000000* 001E4 ;TPA$ACTION
                                                U.237:  .LONG   <<NEXT_RECORD-U.237>-4>                 ;
                                    0000* 001E8 ;TPA$TARGET
                                                U.238:  .WORD   <<U.224-U.238>-2>                       ;
                                    11F2  001EA ;TPA$TYPE
                                                U.239:  .WORD   4594                                    ;
                                    0000* 001EC ;TPA$TARGET
                                                U.240:  .WORD   <<U.224-U.240>-2>                       ;
                                    99F8  001EE ;TPA$TYPE
                                                U.241:  .WORD   -26120                                  ;
                                    0000* 001F0 ;TPA$SUBEXP
                                                U.242:  .WORD   <<U.207-U.242>-2>                       ;
                                00000000V 001F2 ;TPA$ACTION
                                                U.243:  .LONG   <<STORE_SUBS-U.243>-4>                  ;
                                    0000* 001F6 ;TPA$TARGET
                                                U.244:  .WORD   <<SUB_LOOP2-U.244>-2>                   ;
                                    15F6  001F8 ;TPA$TYPE
                                                U.245:  .WORD   5622                                    ;
                                    0000* 001FA ;TPA$TARGET
                                                U.246:  .WORD   <<U.215-U.246>-2>                       ;
                                          001FC ;START_SUBSTRING
                                                U.231:  .BLKB   0
                                    8028  001FC ;TPA$TYPE
                                                U.247:  .WORD   -32728                                  ;
                                00000000V 001FE ;TPA$ACTION
                                                U.248:  .LONG   <<INIT_SUBS-U.248>-4>                   ;
                                    15F6  00202 ;TPA$TYPE
                                                U.249:  .WORD   5622                                    ;
                                    FFFF  00204 ;TPA$TARGET
                                                U.250:  .WORD   -1                                      ;
                                          00206 LEFT_SUBSTRING:
                                                .BLKB   0
                                    99F8  00206 ;TPA$TYPE
                                                U.251:  .WORD   -26120                                  ;
                                    0000* 00208 ;TPA$SUBEXP
                                                U.252:  .WORD   <<U.3-U.252>-2>                         ;
                                00000000* 0020A ;TPA$ACTION
                                                U.253:  .LONG   <<NEXT_RECORD-U.253>-4>                 ;
                                    0000* 0020E ;TPA$TARGET
                                                U.254:  .WORD   <<LEFT_SUBSTRING-U.254>-2>              ;
                                    11F2  00210 ;TPA$TYPE
                                                U.255:  .WORD   4594                                    ;
                                    0000* 00212 ;TPA$TARGET
                                                U.256:  .WORD   <<LEFT_SUBSTRING-U.256>-2>              ;
                                    99F8  00214 ;TPA$TYPE
                                                U.257:  .WORD   -26120                                  ;
                                    0000* 00216 ;TPA$SUBEXP
```

```
                   U.258:   .WORD    <<U.207-U.258>-2>                 ;
00000000V 00218 ;TPA$ACTION
                   U.259:   .LONG    <<STORE_SUBS-U.259>-4>            ;
    0000* 0021C ;TPA$TARGET
                   U.261:   .WORD    <<U.260-U.261>-2>                 ;
    903A  0021E ;TPA$TYPE
                   U.262:   .WORD    -28614                            ;
00000000V 00220 ;TPA$ACTION
                   U.263:   .LONG    <<SUBSTRING_COLON-U.263>-4>       ;
    0000* 00224 ;TPA$TARGET
                   U.264:   .WORD    <<U.212-U.264>-2>                 ;
    15F6  00226 ;TPA$TYPE
                   U.265:   .WORD    5622                              ;
    0000* 00228 ;TPA$TARGET
                   U.266:   .WORD    <<U.215-U.266>-2>                 ;
          0022A ;SUBSTRING2
                   U.260:   .BLKB    0
    99F8  0022A ;TPA$TYPE
                   U.267:   .WORD    -26120                            ;
    0000* 0022C ;TPA$SUBEXP
                   U.268:   .WORD    <<U.3-U.268>-2>                   ;
00000000* 0022E ;TPA$ACTION
                   U.269:   .LONG    <<NEXT_RECORD-U.269>-4>           ;
    0000* 00232 ;TPA$TARGET
                   U.270:   .WORD    <<U.260-U.270>-2>                 ;
    11F2  00234 ;TPA$TYPE
                   U.271:   .WORD    4594                              ;
    0000* 00236 ;TPA$TARGET
                   U.272:   .WORD    <<U.260-U.272>-2>                 ;
    903A  00238 ;TPA$TYPE
                   U.273:   .WORD    -28614                            ;
00000000V 0023A ;TPA$ACTION
                   U.274:   .LONG    <<SUBSTRING_COLON-U.274>-4>       ;
    0000* 0023E ;TPA$TARGET
                   U.275:   .WORD    <<U.212-U.275>-2>                 ;
    15F6  00240 ;TPA$TYPE
                   U.276:   .WORD    5622                              ;
    0000* 00242 ;TPA$TARGET
                   U.277:   .WORD    <<U.17-U.277>-2>                  ;
          00244 ;RIGHT_SUBSTRING
                   U.212:   .BLKB    0
    99F8  00244 ;TPA$TYPE
                   U.278:   .WORD    -26120                            ;
    0000* 00246 ;TPA$SUBEXP
                   U.279:   .WORD    <<U.3-U.279>-2>                   ;
00000000* 00248 ;TPA$ACTION
                   U.280:   .LONG    <<NEXT_RECORD-U.280>-4>           ;
    0000* 0024C ;TPA$TARGET
                   U.281:   .WORD    <<U.212-U.281>-2>                 ;
    11F2  0024E ;TPA$TYPE
                   U.282:   .WORD    4594                              ;
    0000* 00250 ;TPA$TARGET
                   U.283:   .WORD    <<U.212-U.283>-2>                 ;
    99F8  00252 ;TPA$TYPE
                   U.284:   .WORD    -26120                            ;
    0000* 00254 ;TPA$SUBEXP
                   U.285:   .WORD    <<U.207-U.285>-2>                 ;
```

```
                                    00000000V 00256 ;TPA$ACTION
                                              U.286:  .LONG   <<STORE_SUBS-U.286>-4>          ;
                                        0000* 0025A ;TPA$TARGET
                                              U.288:  .WORD   <<U.287-U.288>-2>               ;
                                        9029  0025C ;TPA$TYPE
                                              U.289:  .WORD   -28631                          ;
                                    00000000V 0025E ;TPA$ACTION
                                              U.290:  .LONG   <<END_SUBSTRING-U.290>-4>       ;
                                        FFFF  00262 ;TPA$TARGET
                                              U.291:  .WORD   -1                              ;
                                        15F6  00264 ;TPA$TYPE
                                              U.292:  .WORD   5622                            ;
                                        0000* 00266 ;TPA$TARGET
                                              U.293:  .WORD   <<U.215-U.293>-2>               ;
                                              00268 ;SUBSTRING3
                                              U.287:  .BLKB   0
                                        99F8  00268 ;TPA$TYPE
                                              U.294:  .WORD   -26120                          ;
                                        0000* 0026A ;TPA$SUBEXP
                                              U.295:  .WORD   <<U.3-U.295>-2>                 ;
                                    00000000* 0026C ;TPA$ACTION
                                              U.296:  .LONG   <<NEXT_RECORD-U.296>-4>         ;
                                        0000* 00270 ;TPA$TARGET
                                              U.297:  .WORD   <<U.287-U.297>-2>               ;
                                        11F2  00272 ;TPA$TYPE
                                              U.298:  .WORD   4594                            ;
                                        0000* 00274 ;TPA$TARGET
                                              U.299:  .WORD   <<U.287-U.299>-2>               ;
                                        9029  00276 ;TPA$TYPE
                                              U.300:  .WORD   -28631                          ;
                                    00000000V 00278 ;TPA$ACTION
                                              U.301:  .LONG   <<END_SUBSTRING-U.301>-4>       ;
                                        FFFF  0027C ;TPA$TARGET
                                              U.302:  .WORD   -1                              ;
                                        15F6  0027E ;TPA$TYPE
                                              U.303:  .WORD   5622                            ;
                                        0000* 00280 ;TPA$TARGET
                                              U.304:  .WORD   <<U.17-U.304>-2>                ;
                                              00282 ;IDENTIFIER
                                              U.188:  .BLKB   0
                                        85EE  00282 ;TPA$TYPE
                                              U.305:  .WORD   -31250                          ;
                                    00000000V 00284 ;TPA$ACTION
                                              U.306:  .LONG   <<BLANKS_ON-U.306>-4>           ;
                                        91F1  00288 ;TPA$TYPE
                                              U.307:  .WORD   -28175                          ;
                                    00000000V 0028A ;TPA$ACTION
                                              U.308:  .LONG   <<BLANKS_OFF-U.308>-4>          ;
                                        FFFF  0028E ;TPA$TARGET
                                              U.309:  .WORD   -1                              ;
                                        95F6  00290 ;TPA$TYPE
                                              U.310:  .WORD   -27146                          ;
                                    00000000V 00292 ;TPA$ACTION
                                              U.311:  .LONG   <<BLANKS_OFF-U.311>-4>          ;
                                        FFFF  00296 ;TPA$TARGET
                                              U.312:  .WORD   -1                              ;
                                              00298 ;DECIMAL_INTEGER
```

```
                  U.207:   .BLKB    0
     9DF8  00298  ;TPA$TYPE
                  U.313:   .WORD    -25096                                      ;
    0000*  0029A  ;TPA$SUBEXP
                  U.315:   .WORD    <<U.314-U.315>-2>                            ;
00000000V  0029C  ;TPA$ACTION
                  U.316:   .LONG    <<CONVERT_INTEGER-U.316>-4>                  ;
     FFFF  002A0  ;TPA$TARGET
                  U.317:   .WORD    -1                                          ;
           002A2  ;INTEGER
                  U.314:   .BLKB    0
     802B  002A2  ;TPA$TYPE
                  U.318:   .WORD    -32725                                      ;
00000000V  002A4  ;TPA$ACTION
                  U.319:   .LONG    <<BLANKS_ON-U.319>-4>                        ;
     802D  002A8  ;TPA$TYPE
                  U.320:   .WORD    -32723                                      ;
00000000V  002AA  ;TPA$ACTION
                  U.321:   .LONG    <<BLANKS_ON-U.321>-4>                        ;
     85F6  002AE  ;TPA$TYPE
                  U.322:   .WORD    -31242                                      ;
00000000V  002B0  ;TPA$ACTION
                  U.323:   .LONG    <<BLANKS_ON-U.323>-4>                        ;
     91F3  002B4  ;TPA$TYPE
                  U.324:   .WORD    -28173                                      ;
00000000V  002B6  ;TPA$ACTION
                  U.325:   .LONG    <<BLANKS_OFF-U.325>-4>                       ;
     FFFF  002BA  ;TPA$TARGET
                  U.326:   .WORD    -1                                          ;
     95F6  002BC  ;TPA$TYPE
                  U.327:   .WORD    -27146                                      ;
00000000V  002BE  ;TPA$ACTION
                  U.328:   .LONG    <<BLANKS_OFF-U.328>-4>                       ;
     FFFE  002C2  ;TPA$TARGET
                  U.329:   .WORD    -2                                          ;
           002C4  ;REAL
                  U.170:   .BLKB    0
     802B  002C4  ;TPA$TYPE
                  U.330:   .WORD    -32725                                      ;
00000000V  002C6  ;TPA$ACTION
                  U.331:   .LONG    <<BLANKS_ON-U.331>-4>                        ;
     802D  002CA  ;TPA$TYPE
                  U.332:   .WORD    -32723                                      ;
00000000V  002CC  ;TPA$ACTION
                  U.333:   .LONG    <<BLANKS_ON-U.333>-4>                        ;
     85F6  002D0  ;TPA$TYPE
                  U.334:   .WORD    -31242                                      ;
00000000V  002D2  ;TPA$ACTION
                  U.335:   .LONG    <<BLANKS_ON-U.335>-4>                        ;
           002D6  REAL1:   .BLKB    0
     11EF  002D6  ;TPA$TYPE
                  U.336:   .WORD    4591                                        ;
    0000*  002D8  ;TPA$TARGET
                  U.337:   .WORD    <<REAL1-U.337>-2>                            ;
     002E  002DA  ;TPA$TYPE
                  U.338:   .WORD    46                                          ;
     05F6  002DC  ;TPA$TYPE
```

```
                   U.339:  .WORD    1526                        ;
             002DE REAL2:  .BLKB    0
      11EF   002DE ;TPA$TYPE
                   U.340:  .WORD    4591                        ;
      0000*  002E0 ;TPA$TARGET
                   U.341:  .WORD    <<REAL2-U.341>-2>           ;
      05F6   002E2 ;TPA$TYPE
                   U.342:  .WORD    1526                        ;
             002E4 EXPONENT:
                           .BLKB    0
      0045   002E4 ;TPA$TYPE
                   U.343:  .WORD    69                          ;
      0065   002E6 ;TPA$TYPE
                   U.344:  .WORD    101                         ;
      0044   002E8 ;TPA$TYPE
                   U.345:  .WORD    68                          ;
      0064   002EA ;TPA$TYPE
                   U.346:  .WORD    100                         ;
      0051   002EC ;TPA$TYPE
                   U.347:  .WORD    81                          ;
      0071   002EE ;TPA$TYPE
                   U.348:  .WORD    113                         ;
      05F6   002F0 ;TPA$TYPE
                   U.349:  .WORD    1526                        ;
      002B   002F2 ;TPA$TYPE
                   U.350:  .WORD    43                          ;
      002D   002F4 ;TPA$TYPE
                   U.351:  .WORD    45                          ;
      05F6   002F6 ;TPA$TYPE
                   U.352:  .WORD    1526                        ;
             002F8 EXPONENT2:
                           .BLKB    0
      11EF   002F8 ;TPA$TYPE
                   U.353:  .WORD    4591                        ;
      0000*  002FA ;TPA$TARGET
                   U.354:  .WORD    <<EXPONENT2-U.354>-2>       ;
      05F6   002FC ;TPA$TYPE
                   U.355:  .WORD    1526                        ;
      19F8   002FE ;TPA$TYPE
                   U.356:  .WORD    6648                        ;
      0000*  00300 ;TPA$SUBEXP
                   U.357:  .WORD    <<U.158-U.357>-2>           ;
      FFFE   00302 ;TPA$TARGET
                   U.358:  .WORD    -2                          ;
      15F6   00304 ;TPA$TYPE
                   U.359:  .WORD    5622                        ;
      FFFF   00306 ;TPA$TARGET
                   U.360:  .WORD    -1                          ;
             00308 ;NOT_DELIM
                   U.158:  .BLKB    0
      19F8   00308 ;TPA$TYPE
                   U.361:  .WORD    6648                        ;
      0000*  0030A ;TPA$SUBEXP
                   U.362:  .WORD    <<U.3-U.362>-2>             ;
      FFFE   0030C ;TPA$TARGET
                   U.363:  .WORD    -2                          ;
      11F2   0030E ;TPA$TYPE
```

```
                                                U.364:  .WORD   4594                    ;
                                    FFFE  00310 ;TPA$TARGET
                                                U.365:  .WORD   -2                      ;
                                    102C  00312 ;TPA$TYPE
                                                U.366:  .WORD   4140                    ;
                                    FFFE  00314 ;TPA$TARGET
                                                U.367:  .WORD   -2                      ;
                                    1024  00316 ;TPA$TYPE
                                                U.368:  .WORD   4132                    ;
                                    FFFE  00318 ;TPA$TARGET
                                                U.369:  .WORD   -2                      ;
                                    1026  0031A ;TPA$TYPE
                                                U.370:  .WORD   4134                    ;
                                    FFFE  0031C ;TPA$TARGET
                                                U.371:  .WORD   -2                      ;
                                    1029  0031E ;TPA$TYPE
                                                U.372:  .WORD   4137                    ;
                                    FFFE  00320 ;TPA$TARGET
                                                U.373:  .WORD   -2                      ;
                                    15F6  00322 ;TPA$TYPE
                                                U.374:  .WORD   5622                    ;
                                    FFFF  00324 ;TPA$TARGET
                                                U.375:  .WORD   -1                      ;
                                          00326 ;LOGICAL
                                                U.165:  .BLKB   0
                                    802E  00326 ;TPA$TYPE
                                                U.376:  .WORD   -32722                  ;
                                00000000V 00328 ;TPA$ACTION
                                                U.377:  .LONG   <<BLANKS_ON-U.377>-4>   ;
                                    85F6  0032C ;TPA$TYPE
                                                U.378:  .WORD   -31242                  ;
                                00000000V 0032E ;TPA$ACTION
                                                U.379:  .LONG   <<BLANKS_ON-U.379>-4>   ;
                                    0054  00332 ;TPA$TYPE
                                                U.380:  .WORD   84                      ;
                                    0074  00334 ;TPA$TYPE
                                                U.381:  .WORD   116                     ;
                                    0046  00336 ;TPA$TYPE
                                                U.382:  .WORD   70                      ;
                                    0466  00338 ;TPA$TYPE
                                                U.383:  .WORD   1126                    ;
                                          0033A LOGICAL1:
                                                        .BLKB   0
                                    19F8  0033A ;TPA$TYPE
                                                U.384:  .WORD   6648                    ;
                                    0000* 0033C ;TPA$SUBEXP
                                                U.386:  .WORD   <<U.385-U.386>-2>       ;
                                    0000* 0033E ;TPA$TARGET
                                                U.387:  .WORD   <<LOGICAL1-U.387>-2>    ;
                                    95F6  00340 ;TPA$TYPE
                                                U.388:  .WORD   -27146                  ;
                                00000000V 00342 ;TPA$ACTION
                                                U.389:  .LONG   <<BLANKS_OFF-U.389>-4>  ;
                                    FFFF  00346 ;TPA$TARGET
                                                U.390:  .WORD   -1                      ;
                                          00348 ;LOGICAL2
                                                U.385:  .BLKB   0
```

```
                                                19F8  00348 ;TPA$TYPE
                                                      U.391:  .WORD   6648                                    ;
                                                0000* 0034A ;TPA$SUBEXP
                                                      U.392:  .WORD   <<U.3-U.392>-2>                         ;
                                                FFFE  0034C ;TPA$TARGET
                                                      U.393:  .WORD   -2                                      ;
                                                11F2  0034E ;TPA$TYPE
                                                      U.394:  .WORD   4594                                    ;
                                                FFFE  00350 ;TPA$TARGET
                                                      U.395:  .WORD   -2                                      ;
                                                102C  00352 ;TPA$TYPE
                                                      U.396:  .WORD   4140                                    ;
                                                FFFE  00354 ;TPA$TARGET
                                                      U.397:  .WORD   -2                                      ;
                                                1028  00356 ;TPA$TYPE
                                                      U.398:  .WORD   4136                                    ;
                                                FFFE  00358 ;TPA$TARGET
                                                      U.399:  .WORD   -2                                      ;
                                                103D  0035A ;TPA$TYPE
                                                      U.400:  .WORD   4157                                    ;
                                                FFFE  0035C ;TPA$TARGET
                                                      U.401:  .WORD   -2                                      ;
                                                1024  0035E ;TPA$TYPE
                                                      U.402:  .WORD   4132                                    ;
                                                FFFE  00360 ;TPA$TARGET
                                                      U.403:  .WORD   -2                                      ;
                                                1026  00362 ;TPA$TYPE
                                                      U.404:  .WORD   4134                                    ;
                                                FFFE  00364 ;TPA$TARGET
                                                      U.405:  .WORD   -2                                      ;
                                                15ED  00366 ;TPA$TYPE
                                                      U.406:  .WORD   5613                                    ;
                                                FFFF  00368 ;TPA$TARGET
                                                      U.407:  .WORD   -1                                      ;
                                                      0036A ;COMPLEX
                                                      U.175:  .BLKB   0
                                                1428  0036A ;TPA$TYPE
                                                      U.408:  .WORD   5160                                    ;
                                                0000* 0036C ;TPA$TARGET
                                                      U.410:  .WORD   <<U.409-U.410>-2>                       ;
                                                      0036E ;COMPLEX2
                                                      U.409:  .BLKB   0
                                                99F8  0036E ;TPA$TYPE
                                                      U.411:  .WORD   -26120                                  ;
                                                0000* 00370 ;TPA$SUBEXP
                                                      U.412:  .WORD   <<U.3-U.412>-2>                         ;
                                           00000000* 00372 ;TPA$ACTION
                                                      U.413:  .LONG   <<NEXT_RECORD-U.413>-4>                 ;
                                                0000* 00376 ;TPA$TARGET
                                                      U.414:  .WORD   <<U.409-U.414>-2>                       ;
                                                11F2  00378 ;TPA$TYPE
                                                      U.415:  .WORD   4594                                    ;
                                                0000* 0037A ;TPA$TARGET
                                                      U.416:  .WORD   <<U.409-U.416>-2>                       ;
                                                99F8  0037C ;TPA$TYPE
                                                      U.417:  .WORD   -26120                                  ;
                                                0000* 0037E ;TPA$SUBEXP
```

```
                          U.418:  .WORD   <<U.170-U.418>-2>              ;
00000000V 00380 ;TPA$ACTION
                          U.419:  .LONG   <<STORE_COMPLEX-U.419>-4>      ;
    0000* 00384 ;TPA$TARGET
                          U.421:  .WORD   <<U.420-U.421>-2>              ;
    15F6  00386 ;TPA$TYPE
                          U.422:  .WORD   5622                           ;
    0000* 00388 ;TPA$TARGET
                          U.423:  .WORD   <<U.17-U.423>-2>               ;
          0038A ;COMPLEX3
                          U.420:  .BLKB   0
    99F8  0038A ;TPA$TYPE
                          U.424:  .WORD   -26120                         ;
    0000* 0038C ;TPA$SUBEXP
                          U.425:  .WORD   <<U.3-U.425>-2>                ;
00000000* 0038E ;TPA$ACTION
                          U.426:  .LONG   <<NEXT_RECORD-U.426>-4>        ;
    0000* 00392 ;TPA$TARGET
                          U.427:  .WORD   <<U.420-U.427>-2>              ;
    11F2  00394 ;TPA$TYPE
                          U.428:  .WORD   4594                           ;
    0000* 00396 ;TPA$TARGET
                          U.429:  .WORD   <<U.420-U.429>-2>              ;
    102C  00398 ;TPA$TYPE
                          U.430:  .WORD   4140                           ;
    0000* 0039A ;TPA$TARGET
                          U.432:  .WORD   <<U.431-U.432>-2>              ;
    15F6  0039C ;TPA$TYPE
                          U.433:  .WORD   5622                           ;
    0000* 0039E ;TPA$TARGET
                          U.434:  .WORD   <<U.17-U.434>-2>               ;
          003A0 ;COMPLEX4
                          U.431:  .BLKB   0
    99F8  003A0 ;TPA$TYPE
                          U.435:  .WORD   -26120                         ;
    0000* 003A2 ;TPA$SUBEXP
                          U.436:  .WORD   <<U.3-U.436>-2>                ;
00000000* 003A4 ;TPA$ACTION
                          U.437:  .LONG   <<NEXT_RECORD-U.437>-4>        ;
    0000* 003A8 ;TPA$TARGET
                          U.438:  .WORD   <<U.431-U.438>-2>              ;
    11F2  003AA ;TPA$TYPE
                          U.439:  .WORD   4594                           ;
    0000* 003AC ;TPA$TARGET
                          U.440:  .WORD   <<U.431-U.440>-2>              ;
    99F8  003AE ;TPA$TYPE
                          U.441:  .WORD   -26120                         ;
    0000* 003B0 ;TPA$SUBEXP
                          U.442:  .WORD   <<U.170-U.442>-2>              ;
00000000V 003B2 ;TPA$ACTION
                          U.443:  .LONG   <<STORE_COMPLEX-U.443>-4>      ;
    0000* 003B6 ;TPA$TARGET
                          U.445:  .WORD   <<U.444-U.445>-2>              ;
    15F6  003B8 ;TPA$TYPE
                          U.446:  .WORD   5622                           ;
    0000* 003BA ;TPA$TARGET
                          U.447:  .WORD   <<U.17-U.447>-2>               ;
```

```
                                    003BC ;COMPLEX5
                                          U.444:  .BLKB   0
                                99F8  003BC ;TPA$TYPE
                                          U.448:  .WORD   -26120                          ;
                                0000* 003BE ;TPA$SUBEXP
                                          U.449:  .WORD   <<U.3-U.449>-2>                 ;
                            00000000* 003C0 ;TPA$ACTION
                                          U.450:  .LONG   <<NEXT_RECORD-U.450>-4>         ;
                                0000* 003C4 ;TPA$TARGET
                                          U.451:  .WORD   <<U.444-U.451>-2>               ;
                                11F2  003C6 ;TPA$TYPE
                                          U.452:  .WORD   4594                            ;
                                0000* 003C8 ;TPA$TARGET
                                          U.453:  .WORD   <<U.444-U.453>-2>               ;
                                1029  003CA ;TPA$TYPE
                                          U.454:  .WORD   4137                            ;
                                FFFF  003CC ;TPA$TARGET
                                          U.455:  .WORD   -1                              ;
                                15F6  003CE ;TPA$TYPE
                                          U.456:  .WORD   5622                            ;
                                0000* 003D0 ;TPA$TARGET
                                          U.457:  .WORD   <<U.17-U.457>-2>                ;
                                      003D2 ;CHARACTER
                                          U.179:  .BLKB   0
                                9427  003D2 ;TPA$TYPE
                                          U.458:  .WORD   -27609                          ;
                            00000000V 003D4 ;TPA$ACTION
                                          U.459:  .LONG   <<STRING_OK-U.459>-4>           ;
                                0000* 003D8 ;TPA$TARGET
                                          U.461:  .WORD   <<U.460-U.461>-2>               ;
                                      003DA ;CHARACTER1
                                          U.460:  .BLKB   0
                                91F7  003DA ;TPA$TYPE
                                          U.462:  .WORD   -28169                          ;
                            00000000* 003DC ;TPA$ACTION
                                          U.463:  .LONG   <<NEXT_RECORD-U.463>-4>         ;
                                0000* 003E0 ;TPA$TARGET
                                          U.464:  .WORD   <<U.460-U.464>-2>               ;
                                1027  003E2 ;TPA$TYPE                                                ;
                                          U.465:  .WORD   4135                            ;
                                0000* 003E4 ;TPA$TARGET
                                          U.467:  .WORD   <<U.466-U.467>-2>               ;       ;
                                95ED  003E6 ;TPA$TYPE
                                          U.468:  .WORD   -27155                          ;
                            00000000V 003E8 ;TPA$ACTION
                                          U.469:  .LONG   <<STORE_CHARACTER-U.469>-4>     ;
                                0000* 003EC ;TPA$TARGET
                                          U.470:  .WORD   <<U.460-U.470>-2>               ;
                                      003EE ;NEXT_QUOTE
                                          U.466:  .BLKB   0
                                91F7  003EE ;TPA$TYPE
                                          U.471:  .WORD   -28169                          ;
                            00000000* 003F0 ;TPA$ACTION
                                          U.472:  .LONG   <<NEXT_RECORD-U.472>-4>         ;
                                0000* 003F4 ;TPA$TARGET
                                          U.473:  .WORD   <<U.466-U.473>-2>               ;
                                9027  003F6 ;TPA$TYPE
```

```
                                     U.474:  .WORD    -28633                              ;
               00000000V 003F8 ;TPA$ACTION
                                     U.475:  .LONG    <<STORE_CHARACTER-U.475>-4>         ;
                   0000* 003FC ;TPA$TARGET
                                     U.476:  .WORD    <<U.460-U.476>-2>                   ;
                    15F6 003FE ;TPA$TYPE
                                     U.477:  .WORD    5622                                ;
                    FFFF 00400 ;TPA$TARGET
                                     U.478:  .WORD    -1                                  ;
                         00402 ;ERROR_STATE
                                     U.17:   .BLKB    0
                    91ED 00402 ;TPA$TYPE
                                     U.479:  .WORD    -28179                              ;
               00000000V 00404 ;TPA$ACTION
                                     U.480:  .LONG    <<SYNTAX_ERROR-U.480>-4>            ;
                    FFFE 00408 ;TPA$TARGET
                                     U.481:  .WORD    -2                                  ;
                    95F6 0040A ;TPA$TYPE
                                     U.482:  .WORD    -27146                              ;
               00000000V 0040C ;TPA$ACTION
                                     U.483:  .LONG    <<SYNTAX_ERROR-U.483>-4>            ;
                    FFFE 00410 ;TPA$TARGET
                                     U.484:  .WORD    -2                                  ;
                         00412 ;INVREFVAR_STATE
                                     U.215:  .BLKB    0
                    95F6 00412 ;TPA$TYPE
                                     U.485:  .WORD    -27146                              ;
               00000000V 00414 ;TPA$ACTION
                                     U.486:  .LONG    <<INVREFVAR_ERROR-U.486>-4>         ;
                    FFFE 00418 ;TPA$TARGET
                                     U.487:  .WORD    -2                                  ;

                                             .PSECT   _LIB$KEY0$,NOWRT,  SHR,  PIC,1

                         00000 FOR$$A_NMLKEYWD::
                                             .BLKB    0
                         00000 ;TPA$KEY0
                                     U.1:    .BLKB    0

                                             .EXTRN   FOR$$CVT_TYPE, FOR$$DO_NML_OUTPUT
                                             .EXTRN   FOR$$REC_RSN0, FOR$$REC_WSN0
                                             .EXTRN   FOR$$SIGNAL, FOR$$SIGNAL_STO
                                             .EXTRN   OTS$CVT_TI_L, OTS$CVT_TL_L
                                             .EXTRN   OTS$CVT_T_F, OTS$CVT_T_D
                                             .EXTRN   OTS$CVT_T_G, OTS$CVT_T_H
                                             .EXTRN   LIB$SIG_TO_RET

                                             .PSECT   _FOR$CODE,NOWRT,  SHR,  PIC,2

                    083C 00000 NEXT_RECORD:
                                             .WORD    Save R2,R3,R4,R5,R11                ; 0752
              5B       40  AC  D0 00002      MOVL     64(AP), CCB                         ; 0798
          00000000G    00  16 00006 1$:      JSB      FOR$$REC_RSN0                       ; 0801
                   B0  AB  D6 0000C          INCL     -80(CCB)                            ; 0802
         0C  AC    B0  AB  D0 0000F          MOVL     -80(CCB), 12(AP)                    ; 0803
 08  AC  B4  AB    B0  AB  C3 00014          SUBL3    -80(CCB), -76(CCB), 8(AP)           ; 0804
                   E9  15 00018              BLEQ     1$                                  ; 0806
```

```
                                50          01  D0 0001D           MOVL    #1, R0                                          ; 0807
                                                04 00020           RET                                                     ; 0809

; Routine Size:  33 bytes,    Routine Base:  _FOR$CODE + 0000
```

```
;  750          0810  1 %SBTTL 'INIT_SUBS - Start a subscript/substring '
;  751          0811  1 ROUTINE INIT_SUBS =
;  752          0812  1
;  753          0813  1 !++
;  754          0814  1 ! FUNCTIONAL DESCRIPTION:
;  755          0815  1 !
;  756          0816  1 !       LIB$TPARSE action routine which initiates the evaluation of a subscript
;  757          0817  1 !       or substring.  If the current variable can not have a subscript or
;  758          0818  1 !       a substring, an error routine is called.
;  759          0819  1 !
;  760          0820  1 ! CALLING SEQUENCE:
;  761          0821  1 !
;  762          0822  1 !       status = INIT_SUBS ()
;  763          0823  1 !
;  764          0824  1 ! FORMAL PARAMETERS:
;  765          0825  1 !
;  766          0826  1 !       NONE
;  767          0827  1 !
;  768          0828  1 ! IMPLICIT INPUTS:
;  769          0829  1 !
;  770          0830  1 !       AP      Points to PARAM_BLOCK
;  771          0831  1 !
;  772          0832  1 ! IMPLICIT OUTPUTS:
;  773          0833  1 !
;  774          0834  1 !       PARAM_BLOCK [NML$L_CURIDX] = 0
;  775          0835  1 !
;  776          0836  1 ! COMPLETION STATUS:
;  777          0837  1 !
;  778          0838  1 !       1 for success
;  779          0839  1 !
;  780          0840  1 ! SIDE EFFECTS:
;  781          0841  1 !
;  782          0842  1 !       Can call INVREFVAR_ERROR
;  783          0843  1 !
;  784          0844  1 !--
;  785          0845  1
;  786          0846  2     BEGIN
;  787          0847  2
;  788          0848  2     BUILTIN
;  789          0849  2         AP;                 ! Argument pointer points to parameter block
;  790          0850  2
;  791          0851  2     MAP
;  792          0852  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  793          0853  2
;  794          0854  2     LOCAL
;  795          0855  2         DESCR: REF BLOCK [, BYTE];
;  796          0856  2
;  797          0857  2     DESCR = .AP [NML$A_DESCR];  ! Get descriptor address
;  798          0858  2
;  799          0859  2     !+
;  800          0860  2     ! If this variable is not an array or a string, signal FOR$_INVREFVAR
;  801          0861  2     !-
;  802          0862  2
;  803          0863  3     IF ((.DESCR [DSC$B_CLASS] EQL DSC$K_CLASS_A) OR
;  804          0864  3             (.DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_T))
;  805          0865  2     THEN
;  806          0866  2         AP [NML$L_CURIDX] = 0   ! Set up for start of subscript/substring
```

```
;  807          0867  2        ELSE
;  808          0868  2            CALLG (.AP, INVREFVAR_ERROR);
;  809          0869  2
;  810          0870  2        RETURN 1;
;  811          0871  2
;  812          0872  1        END;
```

```
                                             0000 00000 INIT_SUBS:
                                                                  .WORD   Save nothing                              ; 0811
                       50        3C    AC   D0 00002              MOVL    60(AP), DESCR                             ; 0857
                       04        03    A0   91 00006              CMPB    3(DESCR), #4                              ; 0863
                                       06   13 0000A              BEQL    1$                                        ;
                       0E        02    A0   91 0000C              CMPB    2(DESCR), #14                             ; 0864
                                       05   12 00010              BNEQ    2$                                        ;
                                 48    AC   D4 00012 1$:          CLRL    72(AP)                                    ; 0866
                                       05   11 00015              BRB     3$                                        ;
                 0000V CF              6C   FA 00017 2$:          CALLG   (AP), INVREFVAR_ERROR                     ; 0868
                       50              01   D0 0001C 3$:          MOVL    #1, R0                                    ; 0870
                                            04 0001F              RET                                               ; 0872

; Routine Size:  32 bytes,    Routine Base:  _FOR$CODE + 0021

;  813          0873  1 !<BLF/PAGE>
```

```
;   815          0874  1 %SBTTL 'SUBSTRING_COLON - Mark presence of colon in substring'
;   816          0875  1 ROUTINE SUBSTRING_COLON =
;   817          0876  1
;   818          0877  1 !++
;   819          0878  1 ! FUNCTIONAL DESCRIPTION:
;   820          0879  1 !
;   821          0880  1 !       LIB$TPARSE action routine which is called when a colon is found in
;   822          0881  1 !       a substring.  If no left part has been found, it sets the left part
;   823          0882  1 !       to 1 indicating that the low column was omitted.  If the current
;   824          0883  1 !       variable is not of type CHARACTER, then an error routine is called.
;   825          0884  1 !       If the variable is an array, a subscript must have been previously
;   826          0885  1 !       processed, otherwise an error is given.
;   827          0886  1 !
;   828          0887  1 ! CALLING SEQUENCE:
;   829          0888  1 !
;   830          0889  1 !       status = SUBSTRING_COLON ()
;   831          0890  1 !
;   832          0891  1 ! FORMAL PARAMETERS:
;   833          0892  1 !
;   834          0893  1 !       NONE
;   835          0894  1 !
;   836          0895  1 ! IMPLICIT INPUTS:
;   837          0896  1 !
;   838          0897  1 !       AP      Points to PARAM_BLOCK
;   839          0898  1 !
;   840          0899  1 ! IMPLICIT OUTPUTS:
;   841          0900  1 !
;   842          0901  1 !       If NML$L_CURIDX = 0 then NML$L_CURIDX = 1 and NML$L_SUBSCR[0] = 1
;   843          0902  1 !
;   844          0903  1 ! COMPLETION STATUS:
;   845          0904  1 !
;   846          0905  1 !       1
;   847          0906  1 !
;   848          0907  1 ! SIDE EFFECTS:
;   849          0908  1 !
;   850          0909  1 !       NONE
;   851          0910  1 !
;   852          0911  1 !--
;   853          0912  1
;   854          0913  2     BEGIN
;   855          0914  2
;   856          0915  2     LOCAL
;   857          0916  2         DESCR: REF BLOCK [, BYTE];      ! Address of variable descriptor
;   858          0917  2
;   859          0918  2     BUILTIN
;   860          0919  2         AP;             ! Argument pointer points to parameter block
;   861          0920  2
;   862          0921  2     MAP
;   863          0922  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;   864          0923  2
;   865          0924  2     IF .AP [NML$B_DTYPE] NEQ DSC$K_DTYPE_T
;   866          0925  2     THEN
;   867          0926  2         CALLG (.AP, INVREFVAR_ERROR);   ! Substring not allowed with non-CHARACTER
;   868          0927  2
;   869          0928  2     !+
;   870          0929  2     ! If this variable is an array, then a subscript must have been previously
;   871          0930  2     ! seen for a substring to be valid.
```

```
;  872          0931  2       !-
;  873          0932  2
;  874          0933  2       DESCR = .AP [NML$A_DESCR];
;  875          0934  2       IF .DESCR [DSC$B_CLASS] EQL DSC$K_CLASS_A AND NOT .AP [NML$V_SUBSCRIPT]
;  876          0935  2       THEN
;  877          0936  2           CALLG (.AP, INVREFVAR_ERROR);   ! Substring not allowed with unsubscripted array
;  878          0937  2
;  879          0938  2       IF .AP [NML$L_CURIDX] EQL 0 ! Substring of the form (:n)?
;  880          0939  2       THEN
;  881          0940  3           BEGIN
;  882          0941  3           AP [NML$L_CURIDX] = 1;  ! Left bound is first character
;  883          0942  3           AP [NML$L_SUBSCR] = 1;
;  884          0943  2           END;
;  885          0944  2
;  886          0945  2       AP [NML$V_SUBSTRING] = 1;   ! Indicate substring
;  887          0946  2
;  888          0947  2       RETURN 1;
;  889          0948  2
;  890          0949  1       END;



                                          0000 00000 SUBSTRING_COLON:
                                                               .WORD   Save nothing                       ; 0875
                  0E        44    AC   91 00002                CMPB    68(AP), #14                        ; 0924
                                  05   13 00006                BEQL    1$                                 ;
              0000V   CF          6C   FA 00008                CALLG   (AP), INVREFVAR_ERROR              ; 0926
                  50        3C    AC   D0 0000D 1$:            MOVL    60(AP), DESCR                      ; 0933
                  04        03    A0   91 00011                CMPB    3(DESCR), #4                       ; 0934
                                  0A   12 00015                BNEQ    2$                                 ;
     05        45   AC            03   E0 00017                BBS     #3, 69(AP), 2$                     ;
              0000V   CF          6C   FA 0001C                CALLG   (AP), INVREFVAR_ERROR              ; 0936
                            48    AC   D5 00021 2$:            TSTL    72(AP)                             ; 0938
                                  08   12 00024                BNEQ    3$                                 ;
               48   AC            01   D0 00026                MOVL    #1, 72(AP)                         ; 0941
               4C   AC            01   D0 0002A                MOVL    #1, 76(AP)                         ; 0942
               45   AC            01   88 0002E 3$:            BISB2   #1, 69(AP)                         ; 0945
                    50            01   D0 00032                MOVL    #1, R0                             ; 0947
                                       04 00035                RET                                        ; 0949

; Routine Size:  54 bytes,    Routine Base:  _FOR$CODE + 0041


;  891          0950  1 !<BLF/PAGE>
```

```
;  893          0951  1 %SBTTL 'STORE_SUBS - Store a subscript or substring'
;  894          0952  1 ROUTINE STORE_SUBS =
;  895          0953  1
;  896          0954  1 !++
;  897          0955  1 ! FUNCTIONAL DESCRIPTION:
;  898          0956  1 !
;  899          0957  1 !       LIB$TPARSE action routine which stores the value of a subscript or
;  900          0958  1 !       substring column.  It also checks to see if the allowed number of
;  901          0959  1 !       subscripts have not been exceeded.
;  902          0960  1 !
;  903          0961  1 ! CALLING SEQUENCE:
;  904          0962  1 !
;  905          0963  1 !       status = STORE_SUBS ()
;  906          0964  1 !
;  907          0965  1 ! FORMAL PARAMETERS:
;  908          0966  1 !
;  909          0967  1 !       NONE
;  910          0968  1 !
;  911          0969  1 ! IMPLICIT INPUTS:
;  912          0970  1 !
;  913          0971  1 !       AP      Points to PARAM_BLOCK
;  914          0972  1 !
;  915          0973  1 ! IMPLICIT OUTPUTS:
;  916          0974  1 !
;  917          0975  1 !       PARAM_BLOCK [NML$L_CURIDX] is incremented by 1
;  918          0976  1 !       The value of the subscript is stored in the current subscript vector
;  919          0977  1 !       location.
;  920          0978  1 !
;  921          0979  1 ! COMPLETION STATUS:
;  922          0980  1 !
;  923          0981  1 !       1 for success
;  924          0982  1 !
;  925          0983  1 ! SIDE EFFECTS:
;  926          0984  1 !
;  927          0985  1 !       May call SYNTAX_ERROR
;  928          0986  1 !       May call INVREFVAR_ERROR
;  929          0987  1 !
;  930          0988  1 !--
;  931          0989  1
;  932          0990  2     BEGIN
;  933          0991  2
;  934          0992  2     BUILTIN
;  935          0993  2         AP;             ! Argument pointer points to parameter block
;  936          0994  2
;  937          0995  2     MAP
;  938          0996  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  939          0997  2
;  940          0998  2     LOCAL
;  941          0999  2         SUBSCRIPTS: REF VECTOR [, LONG],
;  942          1000  2         DESCR: REF BLOCK [, BYTE];
;  943          1001  2
;  944          1002  2     SUBSCRIPTS = AP [NML$L_SUBSCR];     ! Address of subscript vector
;  945          1003  2
;  946          1004  2     IF .AP [NML$V_SUBSTRING]
;  947          1005  2     THEN
;  948          1006  3         BEGIN
;  949          1007  3         IF .AP [NML$L_CURIDX] GTR 1     ! Only two substring values allowed!
```

```
;  950          1008  3             THEN
;  951          1009  3                 CALLG (.AP, SYNTAX_ERROR);
;  952          1010  3             IF .AP [TPA$L_NUMBER] LEQ 0      ! Substring column can't be LEQ 0
;  953          1011  3             THEN
;  954          1012  3                 CALLG (.AP, INVREFVAR_ERROR);
;  955          1013  3             END
;  956          1014  2         ELSE
;  957          1015  3             BEGIN
;  958          1016  3             DESCR = .AP [NML$A_DESCR];       ! Get descriptor address
;  959          1017  3             IF .DESCR [DSC$B_CLASS] EQL DSC$K_CLASS_A
;  960          1018  3             THEN
;  961          1019  3                 IF .AP [NML$L_CURIDX] GEQ .DESCR [DSC$B_DIMCT]
;  962          1020  3                 THEN
;  963          1021  3                     CALLG (.AP, INVREFVAR_ERROR);   ! Too many subscripts
;  964          1022  2             END;
;  965          1023  2
;  966          1024  2         SUBSCRIPTS [.AP [NML$L_CURIDX]] = .AP [TPA$L_NUMBER];        ! Store subscript
;  967          1025  2         AP [NML$L_CURIDX] = .AP [NML$L_CURIDX] + 1;
;  968          1026  2
;  969          1027  2         RETURN 1;
;  970          1028  2
;  971          1029  1         END;



                                                0004 00000 STORE_SUBS:
                                                                .WORD   Save R2                                 ; 0952
                                 52        4C    AC   9E 00002          MOVAB   76(AP), SUBSCRIPTS              ; 1002
                                 10        45    AC   E9 00006          BLBC    69(AP), 2$                      ; 1004
                                 01        48    AC   D1 0000A          CMPL    72(AP), #1                      ; 1007
                                                 05   15 0000E          BLEQ    1$                              ;
                            0000V  CF            6C   FA 00010          CALLG   (AP), SYNTAX_ERROR              ; 1009
                                           1C    AC   D5 00015 1$:      TSTL    28(AP)                          ; 1010
                                                 11   11 00018          BRB     3$                              ;
                                 50        3C    AC   D0 0001A 2$:      MOVL    60(AP), DESCR                   ; 1016
                                 04        03    A0   91 0001E          CMPB    3(DESCR), #4                    ; 1017
                                                 0E   12 00022          BNEQ    4$                              ;
      48   AC       0B   A0      08              00   ED 00024          CMPZV   #0, #8, 11(DESCR), 72(AP)       ; 1019
                                                 05   14 0002B 3$:      BGTR    4$                              ;
                            0000V  CF            6C   FA 0002D          CALLG   (AP), INVREFVAR_ERROR           ; 1021
                                 50        48    AC   D0 00032 4$:      MOVL    72(AP), R0                      ; 1024
                               6240        1C    AC   D0 00036          MOVL    28(AP), (SUBSCRIPTS)[R0]        ;
                                           48    AC   D6 0003B          INCL    72(AP)                          ; 1025
                                 50              01   D0 0003E          MOVL    #1, R0                          ; 1027
                                                      04 00041          RET                                     ; 1029

; Routine Size:  66 bytes,    Routine Base:  _FOR$CODE + 0077


;  972          1030  1 !<BLF/PAGE>
```

```
;  974          1031  1 %SBTTL 'END_SUBSCRIPT - End an array subscript'
;  975          1032  1 ROUTINE END_SUBSCRIPT =
;  976          1033  1
;  977          1034  1 !++
;  978          1035  1 ! FUNCTIONAL DESCRIPTION:
;  979          1036  1 !
;  980          1037  1 !       LIB$TPARSE action routine which is called at the end of an array subscript.
;  981          1038  1 !       It calls COMPUTE_INDEX to calculate the starting position in the array.
;  982          1039  1 !
;  983          1040  1 ! CALLING SEQUENCE:
;  984          1041  1 !
;  985          1042  1 !       status = END_SUBSCRIPT ()
;  986          1043  1 !
;  987          1044  1 ! FORMAL PARAMETERS:
;  988          1045  1 !
;  989          1046  1 !       NONE
;  990          1047  1 !
;  991          1048  1 ! IMPLICIT INPUTS:
;  992          1049  1 !
;  993          1050  1 !       AP      Points to PARAM_BLOCK
;  994          1051  1 !
;  995          1052  1 ! IMPLICIT OUTPUTS:
;  996          1053  1 !
;  997          1054  1 !       See COMPUTE_INDEX
;  998          1055  1 !       NML$V_SUBSCRIPT = 1, to indicate subscript processed.
;  999          1056  1 !
; 1000          1057  1 ! COMPLETION STATUS:
; 1001          1058  1 !
; 1002          1059  1 !       1 for success
; 1003          1060  1 !
; 1004          1061  1 ! SIDE EFFECTS:
; 1005          1062  1 !
; 1006          1063  1 !       Signals FOR$_INVREFVAR if a subscript is out of bounds.
; 1007          1064  1 !
; 1008          1065  1 !--
; 1009          1066  1
; 1010          1067  2     BEGIN
; 1011          1068  2
; 1012          1069  2     BUILTIN
; 1013          1070  2         AP;                 ! Argument pointer points to parameter block
; 1014          1071  2
; 1015          1072  2     MAP
; 1016          1073  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
; 1017          1074  2
; 1018          1075  3     IF NOT (CALLG (.AP, COMPUTE_INDEX))
; 1019          1076  2     THEN
; 1020          1077  2         CALLG (.AP, INVREFVAR_ERROR);
; 1021          1078  2
; 1022          1079  2     AP [NML$V_SUBSCRIPT] = 1;   ! Allows substring to follow for arrays
; 1023          1080  2
; 1024          1081  2     RETURN 1;
; 1025          1082  2
; 1026          1083  1     END;
```

```
                              0000 00000 END_SUBSCRIPT:
                                             .WORD    Save nothing                    ; 1032
        0000V  CF          6C  FA 00002      CALLG    (AP), COMPUTE_INDEX             ; 1075
               05          50  E8 00007      BLBS     R0, 1$                          ;
        0000V  CF          6C  FA 0000A      CALLG    (AP), INVREFVAR_ERROR           ; 1077
          45   AC          08  88 0000F 1$:  BISB2    #8, 69(AP)                      ; 1079
               50          01  D0 00013      MOVL     #1, R0                          ; 1081
                               04 00016      RET                                      ; 1083

; Routine Size:  23 bytes,    Routine Base:  _FOR$CODE + 00B9

; 1027          1084  1 !<BLF/PAGE>
```

```
;  1029          1085  1 %SBTTL 'COMPUTE_INDEX - Compute the array index'
;  1030          1086  1 ROUTINE COMPUTE_INDEX =
;  1031          1087  1
;  1032          1088  1 !++
;  1033          1089  1 ! FUNCTIONAL DESCRIPTION:
;  1034          1090  1 !
;  1035          1091  1 !       Routine which computes the starting location for the current
;  1036          1092  1 !       variable based on the array subscripts seen.
;  1037          1093  1 !
;  1038          1094  1 ! CALLING SEQUENCE:
;  1039          1095  1 !
;  1040          1096  1 !       status = COMPUTE_INDEX ()
;  1041          1097  1 !
;  1042          1098  1 ! FORMAL PARAMETERS:
;  1043          1099  1 !
;  1044          1100  1 !       NONE
;  1045          1101  1 !
;  1046          1102  1 ! IMPLICIT INPUTS:
;  1047          1103  1 !
;  1048          1104  1 !       AP      Points to PARAM_BLOCK
;  1049          1105  1 !
;  1050          1106  1 ! IMPLICIT OUTPUTS:
;  1051          1107  1 !
;  1052          1108  1 !       PARAM_BLOCK [NML$A_VARCUR] = Starting address
;  1053          1109  1 !       PARAM_BLOCK [NML$A_VARSTART] = Starting address
;  1054          1110  1 !
;  1055          1111  1 ! COMPLETION STATUS:
;  1056          1112  1 !
;  1057          1113  1 !       1 for success
;  1058          1114  1 !       SS$_SUBRNG for subscript out of range
;  1059          1115  1 !
;  1060          1116  1 ! SIDE EFFECTS:
;  1061          1117  1 !
;  1062          1118  1 !       NONE
;  1063          1119  1 !
;  1064          1120  1 !--
;  1065          1121  1
;  1066          1122  2     BEGIN
;  1067          1123  2
;  1068          1124  2     BUILTIN
;  1069          1125  2         AP;                 ! Argument pointer points to parameter block
;  1070          1126  2
;  1071          1127  2     MAP
;  1072          1128  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1073          1129  2
;  1074          1130  2     LOCAL
;  1075          1131  2         DESCR: REF BLOCK [, BYTE],          ! Variable descriptor
;  1076          1132  2         MULTIPLIERS: REF VECTOR [,LONG],    ! Multiplier array
;  1077          1133  2         LAST_MULT,                          ! Previous bounds' multiplier
;  1078          1134  2         BOUNDS: REF VECTOR [,LONG],         ! Current bounds
;  1079          1135  2         SUBSCRIPT: REF VECTOR [,LONG],      ! Current subscript
;  1080          1136  2         DIMENSION,                          ! Current dimension
;  1081          1137  2         OFFSET;                             ! Offset into array
;  1082          1138  2
;  1083          1139  2     ENABLE
;  1084          1140  2         LIB$SIG_TO_RET; ! Return SS$_SUBRNG as a status
;  1085          1141  2
```

```
;  1086          1142  2     DESCR = .AP [NML$A_DESCR];          ! Get descriptor address
;  1087          1143  2
;  1088          1144  2     !+
;  1089          1145  2     ! If the descriptor class is not ARRAY, then a subscript is illegal.
;  1090          1146  2     !-
;  1091          1147  2
;  1092          1148  2     IF .DESCR [DSC$B_CLASS] NEQ DSC$K_CLASS_A
;  1093          1149  2     THEN
;  1094          1150  2         RETURN 0;
;  1095          1151  2
;  1096          1152  2     !+
;  1097          1153  2     ! If the number of subscripts doesn't match the number of dimensions, then
;  1098          1154  2     ! it is an error.
;  1099          1155  2     !-
;  1100          1156  2
;  1101          1157  2     IF .DESCR [DSC$B_DIMCT] NEQ .AP [NML$L_CURIDX]
;  1102          1158  2     THEN
;  1103          1159  2         RETURN 0;       ! Number of subscripts don't match
;  1104          1160  2
;  1105          1161  2     DIMENSION = .AP [NML$L_CURIDX] - 1;
;  1106          1162  2     SUBSCRIPT = AP [NML$L_SUBSCR] + (4 * .DIMENSION);
;  1107          1163  2     MULTIPLIERS = DESCR [DSC$L_M1] + (4 * .DIMENSION) - 4;
;  1108          1164  2     LAST_MULT = .MULTIPLIERS [0];
;  1109          1165  2     BOUNDS = MULTIPLIERS [2] + (8 * .DIMENSION);
;  1110          1166  2     OFFSET = 0;
;  1111          1167  2
;  1112          1168  2     !+
;  1113          1169  2     ! For each dimension, from last to first, compute the offset into the
;  1114          1170  2     ! array.  If a subscript is out of bounds, the INDEX instruction will
;  1115          1171  2     ! signal an error.
;  1116          1172  2     !-
;  1117          1173  2
;  1118          1174  2     DECR DIM FROM .DIMENSION TO 0 DO
;  1119          1175  3         BEGIN
;  1120          1176  3         IF .DIM EQL 0
;  1121          1177  3         THEN
;  1122          1178  3             LAST_MULT = 1;
;  1123          1179  3         INDEX (SUBSCRIPT [0], BOUNDS [0], BOUNDS [1], LAST_MULT,
;  1124          1180  3                OFFSET, OFFSET);
;  1125          1181  3         MULTIPLIERS = MULTIPLIERS [-1];
;  1126          1182  3         LAST_MULT = .MULTIPLIERS [0];
;  1127          1183  3         BOUNDS = BOUNDS [-2];
;  1128          1184  3         SUBSCRIPT = SUBSCRIPT [-1];
;  1129          1185  2         END;
;  1130          1186  2
;  1131          1187  2     AP [NML$A_VARCUR] = .DESCR [DSC$A_A0] + (.OFFSET * .AP [NML$W_STRIDE]);
;  1132          1188  2     AP [NML$A_VARSTART] = .AP [NML$A_VARCUR];
;  1133          1189  2
;  1134          1190  2     RETURN 1;
;  1135          1191  2
;  1136          1192  1     END;


                                                007C 00000 COMPUTE_INDEX:
```

```
                                                               .WORD   Save R2,R3,R4,R5,R6                           ; 1086
                                6D      0063  CF  DE 00002     MOVAL   5$, (FP)                                      ; 1122
                                53        3C  AC  D0 00007     MOVL    60(AP), DESCR                                 ; 1142
                                04        03  A3  91 0000B     CMPB    3(DESCR), #4                                  ; 1148
                                              55  12 0000F     BNEQ    4$                                            ;
     48  AC      0B  A3         08            00  ED 00011     CMPZV   #0, #8, 11(DESCR), 72(AP)                     ; 1157
                                              4C  12 00018     BNEQ    4$                                            ;
                     50     48  AC            01  C3 0001A     SUBL3   #1, 72(AP), DIMENSION                         ; 1161
                                56     4C AC40  DE 0001F       MOVAL   76(AP)[DIMENSION], SUBSCRIPT                  ; 1162
                                54     10 A340  DE 00024       MOVAL   16(DESCR)[DIMENSION], MULTIPLIERS             ; 1163
                                55            64  D0 00029     MOVL    (MULTIPLIERS), LAST_MULT                      ; 1164
                                52     08 A440  7E 0002C       MOVAQ   8(MULTIPLIERS)[DIMENSION], BOUNDS             ; 1165
                                              51  D4 00031     CLRL    OFFSET                                        ; 1166
                                              50  D6 00033     INCL    DIM                                           ; 1174
                                              16  11 00035     BRB     3$                                            ;
                                              03  12 00037 1$: BNEQ    2$                                            ; 1176
                                55            01  D0 00039     MOVL    #1, LAST_MULT                                 ; 1178
         55      04  A2         62            66  0A 0003C 2$: INDEX   (SUBSCRIPT), (BOUNDS), 4(BOUNDS), -           ; 1179
                                51            51     00042             LAST_MULT, OFFSET, OFFSET                     ;
                                55            74  D0 00044     MOVL    -(MULTIPLIERS), LAST_MULT                     ; 1182
                                52            08  C2 00047     SUBL2   #8, BOUNDS                                    ; 1183
                                56            04  C2 0004A     SUBL2   #4, SUBSCRIPT                                 ; 1184
                                E7            50  F4 0004D 3$: SOBGEQ  DIM, 1$                                       ; 1174
                                50        3A  AC  3C 00050     MOVZWL  58(AP), R0                                    ; 1187
                                51            50  C4 00054     MULL2   R0, R1                                        ;
                           34   AC     10 B341  9E 00057       MOVAB   @16(DESCR)[R1], 52(AP)                        ;
                           2C   AC        34  AC  D0 0005D     MOVL    52(AP), 44(AP)                                ; 1188
                                50            01  D0 00062     MOVL    #1, R0                                        ; 1190
                                                  04 00065     RET                                                   ;
                                              50  D4 00066 4$: CLRL    R0                                            ; 1192
                                                  04 00068     RET                                                   ;
                                            0000 00069 5$:     .WORD   Save nothing                                  ; 1122
                                              7E  D4 0006B     CLRL    -(SP)                                         ;
                                              5E  DD 0006D     PUSHL   SP                                            ;
                                7E        04  AC  7D 0006F     MOVQ    4(AP), -(SP)                                  ;
                     00000000G  00            03  FB 00073     CALLS   #3, LIB$SIG_TO_RET                            ;
                                                  04 0007A     RET                                                   ;

; Routine Size:  123 bytes,    Routine Base:  _FOR$CODE + 00D0

; 1137          1193  1 !<BLF/PAGE>
```

```
;1139     1194  1 %SBTTL 'END_SUBSTRING - End a substring'
;1140     1195  1 ROUTINE END_SUBSTRING =
;1141     1196  1
;1142     1197  1 !++
;1143     1198  1 ! FUNCTIONAL DESCRIPTION:
;1144     1199  1 !
;1145     1200  1 !       LIB$TPARSE action routine which evaluates a substring reference.
;1146     1201  1 !
;1147     1202  1 ! CALLING SEQUENCE:
;1148     1203  1 !
;1149     1204  1 !       status = END_SUBSTRING ()
;1150     1205  1 !
;1151     1206  1 ! FORMAL PARAMETERS:
;1152     1207  1 !
;1153     1208  1 !       NONE
;1154     1209  1 !
;1155     1210  1 ! IMPLICIT INPUTS:
;1156     1211  1 !
;1157     1212  1 !       AP      Points to PARAM_BLOCK
;1158     1213  1 !
;1159     1214  1 ! IMPLICIT OUTPUTS:
;1160     1215  1 !
;1161     1216  1 !       PARAM_BLOCK [NML$A_VARCUR] - Set to starting point
;1162     1217  1 !       PARAM_BLOCK [NML$W_VARSIZE] - Set to string size
;1163     1218  1 !       PARAM_BLOCK [NML$A_VARSTART] - Set to starting point
;1164     1219  1 !
;1165     1220  1 ! COMPLETION STATUS:
;1166     1221  1 !
;1167     1222  1 !       1 for success
;1168     1223  1 !
;1169     1224  1 ! SIDE EFFECTS:
;1170     1225  1 !
;1171     1226  1 !       Can call INVREFVAR_ERROR if the substring is out-of-bounds.
;1172     1227  1 !
;1173     1228  1 !--
;1174     1229  1
;1175     1230  2     BEGIN
;1176     1231  2
;1177     1232  2     BUILTIN
;1178     1233  2         AP;                 ! Argument pointer points to parameter block
;1179     1234  2
;1180     1235  2     MAP
;1181     1236  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;1182     1237  2
;1183     1238  2     IF .AP [NML$L_CURIDX] EQL 1
;1184     1239  2     THEN
;1185     1240  2         AP [NML$L_SUBSTRHI] = .AP [NML$W_VARSIZE]
;1186     1241  2     ELSE IF .AP [NML$L_CURIDX] NEQ 2
;1187     1242  2     THEN
;1188     1243  2         CALLG (.AP, SYNTAX_ERROR);
;1189     1244  2
;1190     1245  2     IF .AP [NML$L_SUBSTRLO] LEQ 0 OR .AP [NML$L_SUBSTRHI] LSS .AP [NML$L_SUBSTRLO] OR
;1191     1246  2        .AP [NML$L_SUBSTRHI] GTR .AP [NML$W_VARSIZE]
;1192     1247  2     THEN
;1193     1248  2         CALLG (.AP, INVREFVAR_ERROR);
;1194     1249  2
;1195     1250  2     AP [NML$A_VARCUR] = .AP [NML$A_VARCUR] + .AP [NML$L_SUBSTRLO] - 1;
```

```
; 1196          1251  2        AP [NML$A_VARSTART] = .AP [NML$A_VARCUR];
; 1197          1252  2        AP [NML$W_VARSIZE] = (.AP [NML$L_SUBSTRHI] - .AP [NML$L_SUBSTRLO]) + 1;
; 1198          1253  2
; 1199          1254  2        RETURN 1;
; 1200          1255  2
; 1201          1256  1        END;



                                                  0000 00000 END_SUBSTRING:
                                                                     .WORD    Save nothing                          ; 1195
                        01      48   AC  D1 00002                    CMPL     72(AP), #1                            ; 1238
                                     07  12 00006                    BNEQ     1$                                    ;
                  50    AC      38   AC  3C 00008                    MOVZWL   56(AP), 80(AP)                        ; 1240
                                     0B  11 0000D                    BRB      2$                                    ;
                        02      48   AC  D1 0000F 1$:                CMPL     72(AP), #2                            ; 1241
                                     05  13 00013                    BEQL     2$                                    ;
               0000V    CF           6C  FA 00015                    CALLG    (AP), SYNTAX_ERROR                    ; 1243
                                4C   AC  D5 0001A 2$:                TSTL     76(AP)                                ; 1245
                                     10  15 0001D                    BLEQ     3$                                    ;
                  4C    AC      50   AC  D1 0001F                    CMPL     80(AP), 76(AP)                        ;
                                     09  19 00024                    BLSS     3$                                    ;
   50   AC     38   AC          10   00  ED 00026                    CMPZV    #0, #16, 56(AP), 80(AP)               ; 1246
                                     05  18 0002D                    BGEQ     4$                                    ;
               0000V    CF           6C  FA 0002F 3$:                CALLG    (AP), INVREFVAR_ERROR                 ; 1248
           50     34    AC      4C   AC  C1 00034 4$:                ADDL3    76(AP), 52(AP), R0                    ; 1250
                  34    AC      FF   A0  9E 0003A                    MOVAB    -1(R0), 52(AP)                        ;
                  2C    AC      34   AC  D0 0003F                    MOVL     52(AP), 44(AP)                        ; 1251
           50     50    AC      4C   AC  C3 00044                    SUBL3    76(AP), 80(AP), R0                    ; 1252
              38  AC            50   01  A1 0004A                    ADDW3    #1, R0, 56(AP)                        ;
                                50   01  D0 0004F                    MOVL     #1, R0                                ; 1254
                                         04 00052                    RET                                            ; 1256

; Routine Size:  83 bytes,    Routine Base:  _FOR$CODE + 014B


; 1202          1257  1 !<BLF/PAGE>
```

```
;  1204     1258  1 %SBTTL 'CONVERT_INTEGER - Convert a decimal integer'
;  1205     1259  1 ROUTINE CONVERT_INTEGER =
;  1206     1260  1
;  1207     1261  1 !++
;  1208     1262  1 ! FUNCTIONAL DESCRIPTION:
;  1209     1263  1 !
;  1210     1264  1 !       LIB$TPARSE action routine which converts the current token to a
;  1211     1265  1 !       longword integer which is stored in TPA$L_NUMBER.  If the conversion
;  1212     1266  1 !       fails, an error is signalled.
;  1213     1267  1 !
;  1214     1268  1 ! CALLING SEQUENCE:
;  1215     1269  1 !
;  1216     1270  1 !       status = CONVERT_INTEGER ()
;  1217     1271  1 !
;  1218     1272  1 ! FORMAL PARAMETERS:
;  1219     1273  1 !
;  1220     1274  1 !       NONE
;  1221     1275  1 !
;  1222     1276  1 ! IMPLICIT INPUTS:
;  1223     1277  1 !
;  1224     1278  1 !       AP      Points to PARAM_BLOCK
;  1225     1279  1 !
;  1226     1280  1 ! IMPLICIT OUTPUTS:
;  1227     1281  1 !
;  1228     1282  1 !       TPA$L_NUMBER gets the binary value of the integer token
;  1229     1283  1 !
;  1230     1284  1 ! COMPLETION STATUS:
;  1231     1285  1 !
;  1232     1286  1 !       SS$_NORMAL if success
;  1233     1287  1 !
;  1234     1288  1 ! SIDE EFFECTS:
;  1235     1289  1 !
;  1236     1290  1 !       May signal FOR$_INPCONERR, input conversion error
;  1237     1291  1 !
;  1238     1292  1 !--
;  1239     1293  1
;  1240     1294  2     BEGIN
;  1241     1295  2
;  1242     1296  2     BUILTIN
;  1243     1297  2         AP;                 ! Argument pointer points to parameter block
;  1244     1298  2
;  1245     1299  2     MAP
;  1246     1300  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1247     1301  2
;  1248     1302  2     IF NOT OTS$CVT_TI_L (AP [TPA$L_TOKENCNT], AP [TPA$L_NUMBER])
;  1249     1303  2     THEN
;  1250     1304  2         CALLG (.AP, INPCONERR_ERROR);
;  1251     1305  2
;  1252     1306  2     RETURN 1;
;  1253     1307  2
;  1254     1308  1     END;


                                        0000 00000 CONVERT_INTEGER:
```

```
                                                           .WORD    Save nothing                               ; 1259
                            1C    AC   9F 00002            PUSHAB   28(AP)                                     ; 1302
                            10    AC   9F 00005            PUSHAB   16(AP)                                     ;
           00000000G  00               02 FB 00008            CALLS    #2, OTS$CVT_TI_L                           ;
                      05               50 E8 0000F            BLBS     R0, 1$                                     ;
               0000V  CF               6C FA 00012            CALLG    (AP), INPCONERR_ERROR                      ; 1304
                      50               01 D0 00017 1$:        MOVL     #1, R0                                     ; 1306
                                          04 0001A            RET                                                 ; 1308

; Routine Size:  27 bytes,    Routine Base:  _FOR$CODE + 019E


; 1255          1309  1 !<BLF/PAGE>
```

```
;  1257          1310  1 %SBTTL 'STRING_OK - Is a string value ok?'
;  1258          1311  1 ROUTINE STRING_OK =
;  1259          1312  1
;  1260          1313  1 !++
;  1261          1314  1 ! FUNCTIONAL DESCRIPTION:
;  1262          1315  1 !
;  1263          1316  1 !       LIB$TPARSE action routine which returns success if the current variable
;  1264          1317  1 !       datatype is CHARACTER.  It also sets TPA$V_BLANKS if successful.
;  1265          1318  1 !       If the datatype is not CHARACTER, INPCONERR is signalled.
;  1266          1319  1 !
;  1267          1320  1 ! CALLING SEQUENCE:
;  1268          1321  1 !
;  1269          1322  1 !       status = STRING_OK ()
;  1270          1323  1 !
;  1271          1324  1 ! FORMAL PARAMETERS:
;  1272          1325  1 !
;  1273          1326  1 !       NONE
;  1274          1327  1 !
;  1275          1328  1 ! IMPLICIT INPUTS:
;  1276          1329  1 !
;  1277          1330  1 !       AP      Points to PARAM_BLOCK
;  1278          1331  1 !
;  1279          1332  1 ! IMPLICIT OUTPUTS:
;  1280          1333  1 !
;  1281          1334  1 !       PARAM_BLOCK [TPA$V_BLANKS] = 1 if successful
;  1282          1335  1 !
;  1283          1336  1 ! COMPLETION STATUS:
;  1284          1337  1 !
;  1285          1338  1 !       1 - success
;  1286          1339  1 !
;  1287          1340  1 ! SIDE EFFECTS:
;  1288          1341  1 !
;  1289          1342  1 !       May signal FOR$_INPCONERR, input conversion error
;  1290          1343  1 !
;  1291          1344  1 !--
;  1292          1345  1
;  1293          1346  2     BEGIN
;  1294          1347  2
;  1295          1348  2     BUILTIN
;  1296          1349  2         AP;                 ! Argument pointer points to parameter block
;  1297          1350  2
;  1298          1351  2     MAP
;  1299          1352  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1300          1353  2
;  1301          1354  2     IF .AP [NML$B_DTYPE] NEQ DSC$K_DTYPE_T
;  1302          1355  2     THEN
;  1303          1356  2         CALLG (.AP, INPCONERR_ERROR);   ! Input conversion error
;  1304          1357  2
;  1305          1358  2     IF .AP [NML$A_VARSTART] GEQA .AP [NML$A_VAREND]
;  1306          1359  2     THEN
;  1307          1360  2         FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
;  1308          1361  2
;  1309          1362  2     AP [TPA$V_BLANKS] = 1;
;  1310          1363  2     AP [NML$B_CONSTYPE] = K_CHARACTER;
;  1311          1364  2     AP [NML$L_CONSBLOCK] = .AP [NML$A_VARSTART];
;  1312          1365  2     RETURN 1;
;  1313          1366  2
```

```
; 1314          1367  1     END;



                                            0000 00000 STRING_OK:
                                                              .WORD   Save nothing                                        ; 1311
                     0E         44     AC   91 00002          CMPB    68(AP), #14                                         ; 1354
                                       05   13 00006          BEQL    1$                                                  ;
              0000V  CF                6C   FA 00008          CALLG   (AP), INPCONERR_ERROR                               ; 1356
                 30  AC         2C     AC   D1 0000D 1$:      CMPL    44(AP), 48(AP)                                      ; 1358
                                       0C   1F 00012          BLSSU   2$                                                  ;
                                28     AC   DD 00014          PUSHL   40(AP)                                              ; 1360
                                       12   DD 00017          PUSHL   #18                                                 ;
          00000000G  00                02   FB 00019          CALLS   #2, FOR$$SIGNAL_STO                                 ;
                 04  AC                01   88 00020 2$:      BISB2   #1, 4(AP)                                           ; 1362
                 46  AC                05   90 00024          MOVB    #5, 70(AP)                                          ; 1363
                 68  AC         2C     AC   D0 00028          MOVL    44(AP), 104(AP)                                     ; 1364
                     50                01   D0 0002D          MOVL    #1, R0                                              ; 1365
                                            04 00030          RET                                                         ; 1367

; Routine Size:  49 bytes,    Routine Base:  _FOR$CODE + 01B9


; 1315          1368  1 !<BLF/PAGE>
```

```
;  1317          1369  1 %SBTTL 'STORE_CHARACTER - Store a character in a string'
;  1318          1370  1 ROUTINE STORE_CHARACTER =
;  1319          1371  1
;  1320          1372  1 !++
;  1321          1373  1 ! FUNCTIONAL DESCRIPTION:
;  1322          1374  1 !
;  1323          1375  1 !       LIB$TPARSE action routine which stores the character at TPA$B_CHAR
;  1324          1376  1 !       at the location referenced by NML$A_VARCUR.  NML$A_VARCUR is then
;  1325          1377  1 !       incremented by 1.  If the character would be stored past the end
;  1326          1378  1 !       of the string, the procedure returns success without storing anything.
;  1327          1379  1 !
;  1328          1380  1 ! CALLING SEQUENCE:
;  1329          1381  1 !
;  1330          1382  1 !       status = STORE_CHARACTER ()
;  1331          1383  1 !
;  1332          1384  1 ! FORMAL PARAMETERS:
;  1333          1385  1 !
;  1334          1386  1 !       NONE
;  1335          1387  1 !
;  1336          1388  1 ! IMPLICIT INPUTS:
;  1337          1389  1 !
;  1338          1390  1 !       AP      Points to PARAM_BLOCK
;  1339          1391  1 !
;  1340          1392  1 ! IMPLICIT OUTPUTS:
;  1341          1393  1 !
;  1342          1394  1 !       NONE
;  1343          1395  1 !
;  1344          1396  1 ! COMPLETION STATUS:
;  1345          1397  1 !
;  1346          1398  1 !       1 for success
;  1347          1399  1 !
;  1348          1400  1 ! SIDE EFFECTS:
;  1349          1401  1 !
;  1350          1402  1 !       NONE
;  1351          1403  1 !
;  1352          1404  1 !--
;  1353          1405  1
;  1354          1406  2     BEGIN
;  1355          1407  2
;  1356          1408  2     BUILTIN
;  1357          1409  2         AP;                 ! Argument pointer points to parameter block
;  1358          1410  2
;  1359          1411  2     MAP
;  1360          1412  2         AP: REF BLOCK [, BYTE]  IELD (NML$FIELDS);
;  1361          1413  2
;  1362          1414  2     IF .AP [NML$A_VARCUR] - .AP [NML$A_VARSTART] GEQA .AP [NML$W_VARSIZE]
;  1363          1415  2     THEN
;  1364          1416  2         RETURN 1;
;  1365          1417  2
;  1366          1418  2     CH$WCHAR_A (.AP [TPA$B_CHAR], AP [NML$A_VARCUR]);
;  1367          1419  2     RETURN 1;
;  1368          1420  2
;  1369          1421  1     END;
```

```
                                        0000 00000 STORE_CHARACTER:
                                                        .WORD   Save nothing                         ; 1370
              50      34   AC      2C   AC   C3 00002   SUBL3   44(AP), 52(AP), R0                   ; 1414
      50     38   AC            10      00   ED 00008   CMPZV   #0, #16, 56(AP), R0                  ;
                                        08   1B 0000E   BLEQU   1$                                   ;
                      34   BC      18   AC   90 00010   MOVB    24(AP), @52(AP)                      ; 1418
                                   34   AC   D6 00015   INCL    52(AP)                               ;
                           50           01   D0 00018 1$:  MOVL  #1, R0                              ; 1419
                                             04 0001B   RET                                          ; 1421

; Routine Size:  28 bytes,    Routine Base:  _FOR$CODE + 01EA


; 1370          1422  1 !<BLF/PAGE>
```

```
;  1372     1423  1 %SBTTL 'END_CHARACTER - End a character string'
;  1373     1424  1 ROUTINE END_CHARACTER =
;  1374     1425  1
;  1375     1426  1 !++
;  1376     1427  1 ! FUNCTIONAL DESCRIPTION:
;  1377     1428  1 !
;  1378     1429  1 !       LIB$TPARSE action routine which is called at the end of a character string value.
;  1379     1430  1 !       It blank fills the string if necessary and advances NML$A_VARSTART and
;  1380     1431  1 !       NML$A_VARCUR.  If the repeat count is greater than 1, multiple copies
;  1381     1432  1 !       are stored.
;  1382     1433  1 !
;  1383     1434  1 ! CALLING SEQUENCE:
;  1384     1435  1 !
;  1385     1436  1 !       status = END_CHARACTER ()
;  1386     1437  1 !
;  1387     1438  1 ! FORMAL PARAMETERS:
;  1388     1439  1 !
;  1389     1440  1 !       NONE
;  1390     1441  1 !
;  1391     1442  1 ! IMPLICIT INPUTS:
;  1392     1443  1 !
;  1393     1444  1 !       AP      Points to PARAM_BLOCK
;  1394     1445  1 !
;  1395     1446  1 ! IMPLICIT OUTPUTS:
;  1396     1447  1 !
;  1397     1448  1 !       NML$A_VARCUR = start of next string
;  1398     1449  1 !       NML$A_VARSTART = start of next string
;  1399     1450  1 !       User variable is modified.
;  1400     1451  1 !       NML$L_REPEATCT <= 1
;  1401     1452  1 !
;  1402     1453  1 ! COMPLETION STATUS:
;  1403     1454  1 !
;  1404     1455  1 !       1 for success
;  1405     1456  1 !
;  1406     1457  1 ! SIDE EFFECTS:
;  1407     1458  1 !
;  1408     1459  1 !       NONE
;  1409     1460  1 !
;  1410     1461  1 !--
;  1411     1462  1
;  1412     1463  2     BEGIN
;  1413     1464  2
;  1414     1465  2     BUILTIN
;  1415     1466  2         AP;             ! Argument pointer points to parameter block
;  1416     1467  2
;  1417     1468  2     MAP
;  1418     1469  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1419     1470  2
;  1420     1471  2     LOCAL
;  1421     1472  2         STRINGSIZE;     ! Size of string constant
;  1422     1473  2
;  1423     1474  2     STRINGSIZE = .AP [NML$A_VARCUR] - .AP [NML$A_VARSTART];
;  1424     1475  2     IF .STRINGSIZE LSSU .AP [NML$W_VARSIZE]
;  1425     1476  2     THEN
;  1426     1477  2         CH$FILL (%C' ', (.AP [NML$W_VARSIZE] - .STRINGSIZE), .AP [NML$A_VARCUR]);
;  1427     1478  2
;  1428     1479  2     !+
```

```
; 1429          1480 2       ! Update the current position in the variable.
; 1430          1481 2       !-
; 1431          1482 2
; 1432          1483 2       IF .AP [NML$W_STRIDE] NEQ 0
; 1433          1484 2       THEN
; 1434          1485 2           AP [NML$A_VARCUR] = .AP [NML$A_VARSTART] + .AP [NML$W_STRIDE]
; 1435          1486 2       ELSE
; 1436          1487 2           AP [NML$A_VARCUR] = .AP [NML$A_VAREND];
; 1437          1488 2
; 1438          1489 2       !+
; 1439          1490 2       ! While repeat count is greater than 1, store multiple copies.
; 1440          1491 2       !-
; 1441          1492 2
; 1442          1493 2       WHILE .AP [NML$L_REPEATCT] GTR 1 DO
; 1443          1494 3           BEGIN
; 1444          1495 3           IF .AP [NML$A_VARCUR] GEQA .AP [NML$A_VAREND]
; 1445          1496 3           THEN
; 1446          1497 3               FOR$$SIGNAL_STO (FOR$K_TOOMANYVAL, .AP [NML$A_VARNAME]);
; 1447          1498 3           CH$MOVE (.AP [NML$W_VARSIZE], .AP [NML$A_VARSTART], .AP [NML$A_VARCUR]);
; 1448          1499 3           AP [NML$A_VARCUR] = .AP [NML$A_VARCUR] + .AP [NML$W_STRIDE];     ! Must be array!
; 1449          1500 3           AP [NML$L_REPEATCT] = .AP [NML$L_REPEATCT] - 1;
; 1450          1501 2           END;
; 1451          1502 2
; 1452          1503 2       AP [NML$A_VARSTART] = .AP [NML$A_VARCUR];
; 1453          1504 2       RETURN 1;
; 1454          1505 2
; 1455          1506 1       END;



                                           007C 00000 END_CHARACTER:
                                                              .WORD   Save R2,R3,R4,R5,R6                        ; 1424
                         56       34  AC   9E 00002           MOVAB   52(AP), R6                                 ; 1474
               50        66       2C  AC   C3 00006           SUBL3   44(AP), (R6), STRINGSIZE
   50     38   AC        10           00   ED 0000B           CMPZV   #0, #16, 56(AP), STRINGSIZE                ; 1475
                                      0F   1B 00011           BLEQU   1$
                         51       38  AC   3C 00013           MOVZWL  56(AP), R1                                 ; 1477
               50        51           50   C3 00017           SUBL3   STRINGSIZE, R1, R0
   50          20        6E           00   2C 0001B           MOVC5   #0, (SP), #32, R0, @0(R6)
                                  00  B6      00020
                                  3A  AC   B5 00022 1$:       TSTW    58(AP)                                     ; 1483
                                      0B   13 00025           BEQL    2$
                         50       3A  AC   3C 00027           MOVZWL  58(AP), R0                                 ; 1485
                         66       2C BC40  9E 0002B           MOVAB   @44(AP)[R0], (R6)
                                      04   11 00030           BRB     3$
                         66       30  AC   D0 00032 2$:       MOVL    48(AP), (R6)                               ; 1487
                         01       78  AC   D1 00036 3$:       CMPL    120(AP), #1                                ; 1493
                                      25   15 0003A           BLEQ    5$
                    30   AC           66   D1 0003C           CMPL    (R6), 48(AP)                               ; 1495
                                      0C   1F 00040           BLSSU   4$
                                  28  AC   DD 00042           PUSHL   40(AP)                                     ; 1497
                                      12   DD 00045           PUSHL   #18
               00000000G 00           02   FB 00047           CALLS   #2, FOR$$SIGNAL_STO
          00   B6     2C BC       38  AC   28 0004E 4$:       MOVC3   56(AP), @44(AP), @0(R6)                    ; 1498
                         50       3A  AC   3C 00055           MOVZWL  58(AP), R0                                 ; 1499
```

```
                  66            50  C0 00059        ADDL2   R0, (R6)              ;
                            78  AC  D7 0005C        DECL    120(AP)               ; 1500
                                D5  11 0005F        BRB     3$                    ; 1493
              2C  AC            66  D0 00061 5$:    MOVL    (R6), 44(AP)          ; 1503
                  50            01  D0 00065        MOVL    #1, R0                ; 1504
                                    04 00068        RET                           ; 1506

; Routine Size:  105 bytes,    Routine Base:  _FOR$CODE + 0206

; 1456          1507  1 !<BLF/PAGE>
```

```
;  1458          1508  1 %SBTTL 'STORE_REAL - Store a real constant'
;  1459          1509  1 ROUTINE STORE_REAL =
;  1460          1510  1
;  1461          1511  1 !++
;  1462          1512  1 ! FUNCTIONAL DESCRIPTION:
;  1463          1513  1 !
;  1464          1514  1 !       LIB$TPARSE action routine which converts the real constant at
;  1465          1515  1 !       TPA$L_TOKENCNT and stores the value in NML$L_CONSBLOCK.
;  1466          1516  1 !
;  1467          1517  1 ! CALLING SEQUENCE:
;  1468          1518  1 !
;  1469          1519  1 !       status = STORE_REAL ()
;  1470          1520  1 !
;  1471          1521  1 ! FORMAL PARAMETERS:
;  1472          1522  1 !
;  1473          1523  1 !       NONE
;  1474          1524  1 !
;  1475          1525  1 ! IMPLICIT INPUTS:
;  1476          1526  1 !
;  1477          1527  1 !       AP      Points to PARAM_BLOCK
;  1478          1528  1 !       TPA$L_TOKENCNT - Descriptor of token
;  1479          1529  1 !
;  1480          1530  1 ! IMPLICIT OUTPUTS:
;  1481          1531  1 !
;  1482          1532  1 !       NML$L_CONSBLOCK set to value of token
;  1483          1533  1 !       NML$B_CONSTYPE set to K_REAL
;  1484          1534  1 !
;  1485          1535  1 ! COMPLETION STATUS:
;  1486          1536  1 !
;  1487          1537  1 !       1 for success
;  1488          1538  1 !       0 if the token is of zero length.  This is because the pattern matches
;  1489          1539  1 !         the null string.
;  1490          1540  1 !
;  1491          1541  1 ! SIDE EFFECTS:
;  1492          1542  1 !
;  1493          1543  1 !       May call INPCONERR_ERROR
;  1494          1544  1 !       May signal FOR$_INVARGFOR
;  1495          1545  1 !
;  1496          1546  1 !--
;  1497          1547  1
;  1498          1548  2     BEGIN
;  1499          1549  2
;  1500          1550  2     BUILTIN
;  1501          1551  2         AP;                 ! Argument pointer points to parameter block
;  1502          1552  2
;  1503          1553  2     MAP
;  1504          1554  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1505          1555  2
;  1506          1556  2     !+
;  1507          1557  2     ! If token is of zero length, then return failure.
;  1508          1558  2     !-
;  1509          1559  2
;  1510          1560  2     IF .AP [TPA$L_TOKENCNT] EQL 0
;  1511          1561  2     THEN
;  1512          1562  2         RETURN 0;
;  1513          1563  2
;  1514          1564  2     !+
```

```
        ! Since the pattern for a real matches a string such as "D123", which
        ! might be an identifier, check for the first character being a letter.
        ! If it is, then store the token, set the value to zero and return.
        ! If we don't do this, an identifier like D99999999999 would get a
        ! conversion error immediately.  No other "real" token can possibly
        ! be an identifier.
        !-

        IF CH$RCHAR (.AP [TPA$L_TOKENPTR]) GEQU %C'A' AND
           CH$RCHAR (.AP [TPA$L_TOKENPTR]) LEQU %C'z'
        THEN
            BEGIN
            AP [NML$L_CONSBLOCK] = 0;        ! Set value to zero
            AP [NML$B_CONSTYPE] = K_INTEGER;
            IF .AP [TPA$L_TOKENCNT] LEQ 31
            THEN
                BEGIN
                LOCAL
                    TOKEN: REF VECTOR [, BYTE];
                TOKEN = AP [NML$T_TOKEN];
                TOKEN [0] = .AP [TPA$L_TOKENCNT];
                CH$MOVE (.AP [TPA$L_TOKENCNT], .AP [TPA$L_TOKENPTR], TOKEN [1]);
                END
            ELSE
                AP [NML$T_TOKEN] = 0;
            RETURN 1;
            END
        ELSE
            AP [NML$T_TOKEN] = 0;

        !+
        ! Depending on the destination type, convert the token appropriately.
        !-

        IF ONE_OF (.AP [NML$B_DTYPE],
            DSC$K_DTYPE_L, DSC$K_DTYPE_W, DSC$K_DTYPE_B, DSC$K_DTYPE_WU,
            DSC$K_DTYPE_LU, DSC$K_DTYPE_D, DSC$K_DTYPE_DC)
        THEN
            BEGIN
            IF NOT OTS$CVT_T_D (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK])
            THEN
                CALLG (.AP, INPCONERR_ERROR);
            END

        ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_F, DSC$K_DTYPE_FC)
        THEN
            BEGIN
            IF NOT OTS$CVT_T_F (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK])
            THEN
                CALLG (.AP, INPCONERR_ERROR);
            END

        ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_G, DSC$K_DTYPE_GC)
        THEN
            BEGIN
            IF NOT OTS$CVT_T_G (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK])
            THEN
```

```
3                CALLG (.AP, INPCONERR_ERROR);
3            END
3
3        ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_H)
2        THEN
3            BEGIN
3            IF NOT OTS$CVT_T_H (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK])
3            THEN
3                CALLG (.AP, INPCONERR_ERROR);
3            END
3
3        ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_T)
2        THEN
3            BEGIN
3            AP [NML$L_CONSBLOCK] = 0;       ! Store zero result
3            CALLG (.AP, INPCONERR_ERROR);
3            END
3
2        ELSE        ! Invalid datatype
3            BEGIN
3            FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
2            END;
2
2
2        AP [NML$B_CONSTYPE] = K_REAL;
2
2        RETURN 1;
2
1        END;
```

```
                                  007C 00000 STORE_REAL:
                                                    .WORD   Save R2,R3,R4,R5,R6                 ; 1509
                 56       10  AC  9E 00002          MOVAB   16(AP), R6                          ; 1560
                              66  D5 00006          TSTL    (R6)                                ;
                              03  12 00008          BNEQ    1$                                  ;
                            00B4  31 0000A          BRW     16$                                 ;
             41  8F       14  BC  91 0000D 1$:      CMPB    @20(AP), #65                        ; 1573
                              28  1F 00012          BLSSU   4$                                  ;
             7A  8F       14  BC  91 00014          CMPB    @20(AP), #122                       ; 1574
                              21  1A 00019          BGTRU   4$                                  ;
                          68  AC  D4 0001B          CLRL    104(AP)                             ; 1577
             46  AC           02  90 0001E          MOVB    #2, 70(AP)                          ; 1578
                 1F           66  D1 00022          CMPL    (R6), #31                           ; 1579
                              0F  14 00025          BGTR    2$                                  ;
                 50       7C  AC  9E 00027          MOVAB   124(AP), TOKEN                      ; 1584
                 60           66  90 0002B          MOVB    (R6), (TOKEN)                       ; 1585
  01   A0    14  BC           66  28 0002E          MOVC3   (R6), @20(AP), 1(TOKEN)             ; 1586
                              03  11 00034          BRB     3$                                  ; 1579
                          7C  AC  94 00036 2$:      CLRB    124(AP)                             ; 1589
                            0081  31 00039 3$:      BRW     15$                                 ; 1590
                          7C  AC  94 0003C 4$:      CLRB    124(AP)                             ; 1593
                 52       44  AC  9A 0003F          MOVZBL  68(AP), R2                          ; 1601
     50 18940000 8F           52  78 00043          ASHL    R2, #462684160, R0                  ;
```

```
                                     11  18 0004B          BGEQ    6$                                      ;
                                68   AC  9F 0004D          PUSHAB  104(AP)                                 ; 1604
                                     56  DD 00050          PUSHL   R6                                      ;
               00000000G  00         02  FB 00052          CALLS   #2, OTS$CVT_T_D                         ;
                          5D         50  E8 00059 5$:      BLBS    R0, 14$                                 ;
                                     4B  11 0005C          BRB     12$                                     ; 1606
                          0A         52  91 0005E 6$:      CMPB    R2, #10                                 ; 1609
                                     05  13 00061          BEQL    7$                                      ;
                          0C         52  91 00063          CMPB    R2, #12                                 ;
                                     0E  12 00066          BNEQ    8$                                      ;
                                68   AC  9F 00068 7$:      PUSHAB  104(AP)                                 ; 1612
                                     56  DD 0006B          PUSHL   R6                                      ;
               00000000G  00         02  FB 0006D          CALLS   #2, OTS$CVT_T_F                         ;
                                     E3  11 00074          BRB     5$                                      ;
                          1B         52  91 00076 8$:      CMPB    R2, #27                                 ; 1617
                                     05  13 00079          BEQL    9$                                      ;
                          1D         52  91 0007B          CMPB    R2, #29                                 ;
                                     0E  12 0007E          BNEQ    10$                                     ;
                                68   AC  9F 00080 9$:      PUSHAB  104(AP)                                 ; 1620
                                     56  DD 00083          PUSHL   R6                                      ;
               00000000G  00         02  FB 00085          CALLS   #2, OTS$CVT_T_G                         ;
                                     CB  11 0008C          BRB     5$                                      ;
                          1C         52  91 0008E 10$:     CMPB    R2, #28                                 ; 1625
                                     0E  12 00091          BNEQ    11$                                     ;
                                68   AC  9F 00093          PUSHAB  104(AP)                                 ; 1628
                                     56  DD 00096          PUSHL   R6                                      ;
               00000000G  00         02  FB 00098          CALLS   #2, OTS$CVT_T_H                         ;
                                     B8  11 0009F          BRB     5$                                      ;
                          0E         52  91 000A1 11$:     CMPB    R2, #14                                 ; 1633
                                     0A  12 000A4          BNEQ    13$                                     ;
                                68   AC  D4 000A6          CLRL    104(AP)                                 ; 1636
                   0000V  CF         6C  FA 000A9 12$:     CALLG   (AP), INPCONERR_ERROR                   ; 1637
                                     09  11 000AE          BRB     14$                                     ; 1633
                                     30  DD 000B0 13$:     PUSHL   #48                                     ; 1642
               00000000G  00         01  FB 000B2          CALLS   #1, FOR$$SIGNAL_STO                     ;
                      46  AC         03  90 000B9 14$:     MOVB    #3, 70(AP)                              ; 1646
                          50         01  D0 000BD 15$:     MOVL    #1, R0                                  ; 1648
                                         04 000C0          RET                                             ;
                                     50  D4 000C1 16$:     CLRL    R0                                      ; 1650
                                         04 000C3          RET                                             ;

; Routine Size:  196 bytes,    Routine Base:  _FOR$CODE + 026F

; 1601          1651  1 !<BLF/PAGE>
```

```
;  1603     1652  1 %SBTTL 'STORE_LOGICAL - Store a logical value'
;  1604     1653  1 ROUTINE STORE_LOGICAL =
;  1605     1654  1
;  1606     1655  1 !++
;  1607     1656  1 ! FUNCTIONAL DESCRIPTION:
;  1608     1657  1 !
;  1609     1658  1 !       LIBSTPARSE action routine which converts the logical value at
;  1610     1659  1 !       TPA$L_TOKENCNT and stores the value at NML$L_CONSBLOCK.  If the
;  1611     1660  1 !       token is possibly an identifier, the token is saved at NML$T_TOKEN.
;  1612     1661  1 !
;  1613     1662  1 ! CALLING SEQUENCE:
;  1614     1663  1 !
;  1615     1664  1 !       status = STORE_LOGICAL ()
;  1616     1665  1 !
;  1617     1666  1 ! FORMAL PARAMETERS:
;  1618     1667  1 !
;  1619     1668  1 !       NONE
;  1620     1669  1 !
;  1621     1670  1 ! IMPLICIT INPUTS:
;  1622     1671  1 !
;  1623     1672  1 !       AP      Points to PARAM_BLOCK
;  1624     1673  1 !       TPA$L_TOKENCNT is descriptor of token
;  1625     1674  1 !
;  1626     1675  1 ! IMPLICIT OUTPUTS:
;  1627     1676  1 !
;  1628     1677  1 !       NML$L_CONSBLOCK gets converted value
;  1629     1678  1 !       NML$B_CONSTYPE gets K_LOGICAL
;  1630     1679  1 !       NML$T_TOKEN gets token if possibly an identifier
;  1631     1680  1 !
;  1632     1681  1 ! COMPLETION STATUS:
;  1633     1682  1 !
;  1634     1683  1 !       1 for success
;  1635     1684  1 !
;  1636     1685  1 ! SIDE EFFECTS:
;  1637     1686  1 !
;  1638     1687  1 !       NONE
;  1639     1688  1 !
;  1640     1689  1 !--
;  1641     1690  1
;  1642     1691  2     BEGIN
;  1643     1692  2
;  1644     1693  2     BUILTIN
;  1645     1694  2         AP;                 ! Argument pointer points to parameter block
;  1646     1695  2
;  1647     1696  2     MAP
;  1648     1697  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1649     1698  2
;  1650     1699  2     IF CH$RCHAR (.AP [TPA$L_TOKENPTR]) NEQ %C'.' AND
;  1651     1700  2        .AP [TPA$L_TOKENCNT] LEQ 31
;  1652     1701  2     THEN
;  1653     1702  3         BEGIN
;  1654     1703  3         LOCAL
;  1655     1704  3             TOKEN: REF VECTOR [, BYTE];
;  1656     1705  3         TOKEN = AP [NML$T_TOKEN];
;  1657     1706  3         TOKEN [0] = .AP [TPA$L_TOKENCNT];
;  1658     1707  3         CH$MOVE (.AP [TPA$L_TOKENCNT], .AP [TPA$L_TOKENPTR], TOKEN [1]);
;  1659     1708  3         END
```

```
; 1660          1709  2      ELSE
; 1661          1710  2          AP [NML$T_TOKEN] = 0;
; 1662          1711  2
; 1663          1712  2      OTS$CVT_TL_L (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK]);
; 1664          1713  2      AP [NML$B_CONSTYPE] = K_LOGICAL;
; 1665          1714  2      RETURN 1;
; 1666          1715  2
; 1667          1716  1      END;



                                          003C 00000 STORE_LOGICAL:
                                                          .WORD   Save R2,R3,R4,R5                          ; 1653
                    2E       14    BC  91 00002          CMPB    @20(AP), #46                              ; 1699
                                   17  13 00006          BEQL    1$                                        ;
                    1F       10    AC  D1 00008          CMPL    16(AP), #31                               ; 1700
                                   11  14 0000C          BGTR    1$                                        ;
                    50       7C    AC  9E 0000E          MOVAB   124(AP), TOKEN                            ; 1705
                    60       10    AC  90 00012          MOVB    16(AP), (TOKEN)                           ; 1706
        01   A0       14  BC       10    AC  28 00016          MOVC3   16(AP), @20(AP), 1(TOKEN)                 ; 1707
                                   03  11 0001D          BRB     2$                                        ; 1699
                             7C    AC  94 0001F 1$:      CLRB    124(AP)                                   ; 1710
                             68    AC  9F 00022 2$:      PUSHAB  104(AP)                                   ; 1712
                             10    AC  9F 00025          PUSHAB  16(AP)                                    ;
            00000000G  00          02  FB 00028          CALLS   #2, OTS$CVT_TL_L                          ;
                   46  AC          01  90 0002F          MOVB    #1, 70(AP)                                ; 1713
                    50             01  D0 00033          MOVL    #1, R0                                    ; 1714
                                       04 00036          RET                                               ; 1716

; Routine Size:  55 bytes,    Routine Base:  _FOR$CODE + 0333


; 1668          1717  1 !<BLF/PAGE>
```

```
; 1670          1718  1 %SBTTL 'STORE_COMPLEX - Store a complex constant'
; 1671          1719  1 ROUTINE STORE_COMPLEX =
; 1672          1720  1
; 1673          1721  1 !++
; 1674          1722  1 ! FUNCTIONAL DESCRIPTION:
; 1675          1723  1 !
; 1676          1724  1 !       LIB$TPARSE action routine which converts the current token as a real
; 1677          1725  1 !       value and converts it to either the real part or the imaginary part
; 1678          1726  1 !       of a complex value.
; 1679          1727  1 !
; 1680          1728  1 ! CALLING SEQUENCE:
; 1681          1729  1 !
; 1682          1730  1 !       status = STORE_COMPLEX ()
; 1683          1731  1 !
; 1684          1732  1 ! FORMAL PARAMETERS:
; 1685          1733  1 !
; 1686          1734  1 !       NONE
; 1687          1735  1 !
; 1688          1736  1 ! IMPLICIT INPUTS:
; 1689          1737  1 !
; 1690          1738  1 !       AP      Points to PARAM_BLOCK
; 1691          1739  1 !       TPA$L_TOKENCNT - Descriptor of token
; 1692          1740  1 !       NML$V_IMAG     - Set if real part already seen
; 1693          1741  1 !
; 1694          1742  1 ! IMPLICIT OUTPUTS:
; 1695          1743  1 !
; 1696          1744  1 !       NML$L_CONSBLOCK set to value of token
; 1697          1745  1 !       NML$B_CONSTYPE set to K_COMPLEX
; 1698          1746  1 !       NML$V_IMAG set to 1
; 1699          1747  1 !
; 1700          1748  1 ! COMPLETION STATUS:
; 1701          1749  1 !
; 1702          1750  1 !       1 for success
; 1703          1751  1 !       0 if the token is of zero length.  This is because the pattern matches
; 1704          1752  1 !         the null string.
; 1705          1753  1 !
; 1706          1754  1 ! SIDE EFFECTS:
; 1707          1755  1 !
; 1708          1756  1 !       May call INPCONERR_ERROR
; 1709          1757  1 !       May signal FOR$_INVARGFOR
; 1710          1758  1 !
; 1711          1759  1 !--
; 1712          1760  1
; 1713          1761  2     BEGIN
; 1714          1762  2
; 1715          1763  2     BUILTIN
; 1716          1764  2         AP;                 ! Argument pointer points to parameter block
; 1717          1765  2
; 1718          1766  2     MAP
; 1719          1767  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
; 1720          1768  2
; 1721          1769  2     LOCAL
; 1722          1770  2         L_DTYPE,                            ! Local data type
; 1723          1771  2         L_CONSBLOCK: VECTOR [4, LONG];  ! Local constant block
; 1724          1772  2
; 1725          1773  2     !+
; 1726          1774  2     ! If token is of zero length, then return failure.
```

```
;  1727          1775  2      !-
;  1728          1776  2
;  1729          1777  2      IF .AP [TPA$L_TOKENCNT] EQL 0
;  1730          1778  2      THEN
;  1731          1779  2          RETURN 0;
;  1732          1780  2
;  1733          1781  2      !+
;  1734          1782  2      ! Depending on the destination type, convert the token appropriately.
;  1735          1783  2      !-
;  1736          1784  2
;  1737          1785  2
;  1738          1786  3      IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_F, DSC$K_DTYPE_FC)
;  1739          1787  2      THEN
;  1740          1788  3          BEGIN
;  1741          1789  3          IF NOT OTS$CVT_T_F (AP [TPA$L_TOKENCNT], L_CONSBLOCK)
;  1742          1790  3          THEN
;  1743          1791  3              CALLG (.AP, INPCONERR_ERROR);
;  1744          1792  3          END
;  1745          1793  3
;  1746        P 1794  2      ELSE IF ONE_OF (.AP [NML$B_DTYPE],
;  1747        P 1795  2          DSC$K_DTYPE_L, DSC$K_DTYPE_W, DSC$K_DTYPE_B, DSC$K_DTYPE_LU,
;  1748          1796  3          DSC$K_DTYPE_WU, DSC$R_DTYPE_D, DSC$R_DTYPE_DC)
;  1749          1797  2      THEN
;  1750          1798  3          BEGIN
;  1751          1799  3          IF NOT OTS$CVT_T_D (AP [TPA$L_TOKENCNT], L_CONSBLOCK)
;  1752          1800  3          THEN
;  1753          1801  3              CALLG (.AP, INPCONERR_ERROR);
;  1754          1802  3          END
;  1755          1803  3
;  1756          1804  3      ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_G, DSC$K_DTYPE_GC)
;  1757          1805  2      THEN
;  1758          1806  3          BEGIN
;  1759          1807  3          IF NOT OTS$CVT_T_G (AP [TPA$L_TOKENCNT], L_CONSBLOCK)
;  1760          1808  3          THEN
;  1761          1809  3              CALLG (.AP, INPCONERR_ERROR);
;  1762          1810  3          END
;  1763          1811  3
;  1764          1812  3      ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_H)
;  1765          1813  2      THEN
;  1766          1814  3          BEGIN
;  1767          1815  3          IF NOT OTS$CVT_T_H (AP [TPA$L_TOKENCNT], L_CONSBLOCK)
;  1768          1816  3          THEN
;  1769          1817  3              CALLG (.AP, INPCONERR_ERROR);
;  1770          1818  3          END
;  1771          1819  3
;  1772          1820  3      ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_T)
;  1773          1821  2      THEN
;  1774          1822  3          BEGIN
;  1775          1823  3          L_CONSBLOCK [0] = 0;     ! Store zero result
;  1776          1824  3          CALLG (.AP, INPCONERR_ERROR);
;  1777          1825  3          END
;  1778          1826  3
;  1779          1827  2      ELSE
;  1780          1828  3          BEGIN
;  1781          1829  3          FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
;  1782          1830  2          END;
;  1783          1831  2
```

```
;  1784          1832  2
;  1785          1833  2      AP [NML$B_CONSTYPE] = K_COMPLEX;
;  1786          1834  2
;  1787          1835  2      !+
;  1788          1836  2      ! Now convert the local constant to the proper complex type and store in
;  1789          1837  2      ! either the real or imaginary part of NML$L_CONSBLOCK.
;  1790          1838  2      !-
;  1791          1839  2
;  1792          1840  2      SELECTONE .AP [NML$B_DTYPE] OF
;  1793          1841  2          SET
;  1794          1842  2          [DSC$K_DTYPE_FC]:
;  1795          1843  2              L_DTYPE = DSC$K_DTYPE_F;
;  1796          1844  2          [DSC$K_DTYPE_DC]:
;  1797          1845  2              L_DTYPE = DSC$K_DTYPE_D;
;  1798          1846  2          [DSC$K_DTYPE_GC]:
;  1799          1847  2              L_DTYPE = DSC$K_DTYPE_G;
;  1800          1848  2          [OTHERWISE]:
;  1801          1849  2              L_DTYPE = .AP [NML$B_DTYPE];
;  1802          1850  2          TES;
;  1803          1851  2      IF NOT .AP [NML$V_IMAG]              ! If real part
;  1804          1852  2      THEN
;  1805          1853  3          BEGIN
;  1806          1854  3          IF NOT FOR$$CVT_TYPE (K_REAL, L_CONSBLOCK,
;  1807          1855  3                                .L_DTYPE, AP [NML$L_CONSBLOCK], 0)
;  1808          1856  3          THEN
;  1809          1857  4              BEGIN
;  1810          1858  4              AP [NML$L_CONSBLOCK] = 0;    ! Store zero result
;  1811          1859  4              CALLG (.AP, INPCONERR_ERROR);
;  1812          1860  3              END;
;  1813          1861  3          AP [NML$V_IMAG] = 1;
;  1814          1862  3          END
;  1815          1863  2      ELSE
;  1816          1864  3          BEGIN
;  1817          1865  3          IF .L_DTYPE EQL DSC$K_DTYPE_H
;  1818          1866  3          THEN
;  1819          1867  3              RETURN 1;
;  1820          1868  3          IF NOT FOR$$CVT_TYPE (K_REAL, L_CONSBLOCK,
;  1821          1869  3                                .L_DTYPE,
;  1822          1870  4                                (IF .L_DTYPE EQL DSC$K_DTYPE_F
;  1823          1871  4                                 THEN
;  1824          1872  4                                     AP [NML$L_CONSBLOCK] + 4
;  1825          1873  4                                 ELSE
;  1826          1874  3                                     AP [NML$L_CONSBLOCK] + 8),
;  1827          1875  3                                 0)
;  1828          1876  3          THEN
;  1829          1877  4              BEGIN
;  1830          1878  4              IF .L_DTYPE EQL DSC$K_DTYPE_F
;  1831          1879  4              THEN
;  1832          1880  4                  AP [NML$L_CONSBLOCK]+4 = 0       ! Store zero result
;  1833          1881  4              ELSE
;  1834          1882  4                  AP [NML$L_CONSBLOCK]+8 = 0;
;  1835          1883  4              CALLG (.AP, INPCONERR_ERROR);
;  1836          1884  3              END;
;  1837          1885  2          END;
;  1838          1886  2
;  1839          1887  2      RETURN 1;
;  1840          1888  2
```

```
; 1841          1889  1      END;



                                        003C 00000 STORE_COMPLEX:
                                                             .WORD   Save R2,R3,R4,R5                                    ; 1719
                        55     0000V   CF   9E 00002         MOVAB   INPCONERR_ERROR, R5
                        54 00000000G   00   9E 00007         MOVAB   FOR$$CVT_TYPE, R4
                        5E             10   C2 0000E         SUBL2   #16, SP
                        53        10   AC   9E 00011         MOVAB   16(AP), R3                                          ; 1777
                                       63   D5 00015         TSTL    (R3)
                                       03   12 00017         BNEQ    1$
                                     00FA   31 00019         BRW     23$
                        52        44   AC   9A 0001C 1$:     MOVZBL  68(AP), R2                                          ; 1786
                        0A             52   91 00020         CMPB    R2, #10
                                       05   13 00023         BEQL    2$
                        0C             52   91 00025         CMPB    R2, #12
                                       10   12 00028         BNEQ    4$
                                4008   8F   BB 0002A 2$:     PUSHR   #^M<R3,SP>                                          ; 1789
             00000000G  00             02   FB 0002E         CALLS   #2, OTS$CVT_T_F
                        57             50   E8 00035 3$:     BLBS    R0, 11$
                                       47   11 00038         BRB     9$                                                  ; 1791
        50 1B940000     8F             52   78 0003A 4$:     ASHL    R2, #462684160, R0                                  ; 1796
                                       0D   18 00042         BGEQ    5$
                                4008   8F   BB 00044         PUSHR   #^M<R3,SP>                                          ; 1799
             00000000G  00             02   FB 00048         CALLS   #2, OTS$CVT_T_D
                                       E4   11 0004F         BRB     3$
                        1B             52   91 00051 5$:     CMPB    R2, #27                                             ; 1804
                                       05   13 00054         BEQL    6$
                        1D             52   91 00056         CMPB    R2, #29
                                       0D   12 00059         BNEQ    7$
                                4008   8F   BB 0005B 6$:     PUSHR   #^M<R3,SP>                                          ; 1807
             00000000G  00             02   FB 0005F         CALLS   #2, OTS$CVT_T_G
                                       CD   11 00066         BRB     3$
                        1C             52   91 00068 7$:     CMPB    R2, #28                                             ; 1812
                                       0D   12 0006B         BNEQ    8$
                                4008   8F   BB 0006D         PUSHR   #^M<R3,SP>                                          ; 1815
             00000000G  00             02   FB 00071         CALLS   #2, OTS$CVT_T_H
                                       BB   11 00078         BRB     3$
                        0E             52   91 0007A 8$:     CMPB    R2, #14                                             ; 1820
                                       07   12 0007D         BNEQ    10$
                                       6E   D4 0007F         CLRL    L_CONSBLOCK                                         ; 1823
                        65             6C   FA 00081 9$:     CALLG   (AP), INPCONERR_ERROR                               ; 1824
                                       09   11 00084         BRB     11$                                                 ; 1820
                                       30   DD 00086 10$:    PUSHL   #48                                                 ; 1829
             00000000G  00             01   FB 00088         CALLS   #1, FOR$$SIGNAL_STO
                    46  AC             04   90 0008F 11$:    MOVB    #4, 70(AP)                                          ; 1833
                        50        44   AC   9A 00093         MOVZBL  68(AP), R0                                          ; 1840
                        0C             50   91 00097         CMPB    R0, #12                                             ; 1842
                                       05   12 0009A         BNEQ    12$
                        52             0A   D0 0009C         MOVL    #10, L_DTYPE                                        ; 1843
                                       17   11 0009F         BRB     15$
                        0D             50   91 000A1 12$:    CMPB    R0, #13                                             ; 1844
                                       05   12 000A4         BNEQ    13$
                        52             0B   D0 000A6         MOVL    #11, L_DTYPE                                        ; 1845
```

```
                                  0D  11 000A9         BRB     15$                              ;
                    1D            50  91 000AB 13$:    CMPB    R0, #29                          ; 1846
                                  05  12 000AE         BNEQ    14$                              ;
                    52            1B  D0 000B0         MOVL    #27, L_DTYPE                     ; 1847
                                  03  11 000B3         BRB     15$                              ;
                    52            50  D0 000B5 14$:    MOVL    R0, L_DTYPE                      ; 1849
        1E     45   AC            01  E0 000B8 15$:    BBS     #1, 69(AP), 17$                  ; 1851
                                  7E  D4 000BD         CLRL    -(SP)                            ; 1855
                          68      AC  9F 000BF         PUSHAB  104(AP)                          ;
                                  52  DD 000C2         PUSHL   L_DTYPE                          ;
                          0C      AE  9F 000C4         PUSHAB  L_CONSBLOCK                      ; 1854
                                  03  DD 000C7         PUSHL   #3                               ; 1855
                    64            05  FB 000C9         CALLS   #5, FOR$$CVT_TYPE                ;
                    06            50  E8 000CC         BLBS    R0, 16$                          ;
                          68      AC  D4 000CF         CLRL    104(AP)                          ; 1858
                    65            6C  FA 000D2         CALLG   (AP), INPCONERR_ERROR            ; 1859
               45   AC            02  88 000D5 16$:    BISB2   #2, 69(AP)                       ; 1861
                                  37  11 000D9         BRB     22$                              ; 1851
                    1C            52  D1 000DB 17$:    CMPL    L_DTYPE, #28                     ; 1865
                                  32  13 000DE         BEQL    22$                              ;
                                  7E  D4 000E0         CLRL    -(SP)                            ; 1868
                                  53  D4 000E2         CLRL    R3                               ; 1870
                    0A            52  D1 000E4         CMPL    L_DTYPE, #10                     ;
                                  08  12 000E7         BNEQ    18$                              ;
                                  53  D6 000E9         INCL    R3                               ;
                    50    6C      AC  9E 000EB         MOVAB   108(AP), R0                      ; 1872
                                  04  11 000EF         BRB     19$                              ;
                    50    70      AC  9E 000F1 18$:    MOVAB   112(AP), R0                      ; 1874
                                  50  DD 000F5 19$:    PUSHL   R0                               ;
                                  52  DD 000F7         PUSHL   L_DTYPE                          ; 1869
                          0C      AE  9F 000F9         PUSHAB  L_CONSBLOCK                      ; 1868
                                  03  DD 000FC         PUSHL   #3                               ;
                    64            05  FB 000FE         CALLS   #5, FOR$$CVT_TYPE                ;
                    0E            50  E8 00101         BLBS    R0, 22$                          ;
                    05            53  E9 00104         BLBC    R3, 20$                          ; 1878
                          6C      AC  D4 00107         CLRL    108(AP)                          ; 1880
                                  03  11 0010A         BRB     21$                              ;
                          70      AC  D4 0010C 20$:    CLRL    112(AP)                          ; 1882
                    65            6C  FA 0010F 21$:    CALLG   (AP), INPCONERR_ERROR            ; 1883
                    50            01  D0 00112 22$:    MOVL    #1, R0                           ; 1887
                                      04 00115         RET                                      ;
                                  50  D4 00116 23$:    CLRL    R0                               ; 1889
                                      04 00118         RET                                      ;

; Routine Size:  281 bytes,    Routine Base:  _FOR$CODE + 036A

; 1842          1890  1 !<BLF/PAGE>
```

```
;  1844          1891  1 %SBTTL 'STORE_REPEAT - Store a repeat count'
;  1845          1892  1 ROUTINE STORE_REPEAT =
;  1846          1893  1
;  1847          1894  1 !++
;  1848          1895  1 ! FUNCTIONAL DESCRIPTION:
;  1849          1896  1 !
;  1850          1897  1 !       LIB$TPARSE action routine which stores the repeat count into the
;  1851          1898  1 !       parameter block.
;  1852          1899  1 !
;  1853          1900  1 ! CALLING SEQUENCE:
;  1854          1901  1 !
;  1855          1902  1 !       status = STORE_REPEAT ()
;  1856          1903  1 !
;  1857          1904  1 ! FORMAL PARAMETERS:
;  1858          1905  1 !
;  1859          1906  1 !       NONE
;  1860          1907  1 !
;  1861          1908  1 ! IMPLICIT INPUTS:
;  1862          1909  1 !
;  1863          1910  1 !       AP      Points to PARAM_BLOCK
;  1864          1911  1 !
;  1865          1912  1 ! IMPLICIT OUTPUTS:
;  1866          1913  1 !
;  1867          1914  1 !       NML$L_REPEATCT gets the repeat count
;  1868          1915  1 !       NML$B_CONSTYPE = K_NULL
;  1869          1916  1 !
;  1870          1917  1 ! COMPLETION STATUS:
;  1871          1918  1 !
;  1872          1919  1 !       1 for success
;  1873          1920  1 !
;  1874          1921  1 ! SIDE EFFECTS:
;  1875          1922  1 !
;  1876          1923  1 !       May signal FOR$_SYNERRNAM, syntax error in NAMELIST input
;  1877          1924  1 !
;  1878          1925  1 !--
;  1879          1926  1
;  1880          1927  2     BEGIN
;  1881          1928  2
;  1882          1929  2     BUILTIN
;  1883          1930  2         AP;                 ! Argument pointer points to parameter block
;  1884          1931  2
;  1885          1932  2     MAP
;  1886          1933  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1887          1934  2
;  1888          1935  2     IF .AP [TPA$L_NUMBER] LEQ 0
;  1889          1936  2     THEN
;  1890          1937  2         CALLG (.AP, SYNTAX_ERROR);
;  1891          1938  2
;  1892          1939  2     AP [NML$L_REPEATCT] = .AP [TPA$L_NUMBER];
;  1893          1940  2     AP [NML$B_CONSTYPE] = K_NULL;       ! Initially treat as null value
;  1894          1941  2
;  1895          1942  2     RETURN 1;
;  1896          1943  2
;  1897          1944  1     END;
```

```
                                                0000 00000 STORE_REPEAT:
                                                           .WORD   Save nothing                    ; 1892
                                          1C    AC  D5 00002         TSTL    28(AP)                ; 1935
                                                05  14 00005         BGTR    1$                    ;
                                    0000V CF          6C  FA 00007   CALLG   (AP), SYNTAX_ERROR     ; 1937
                                      78  AC    1C    AC  D0 0000C 1$: MOVL    28(AP), 120(AP)       ; 1939
                                          46    AC  94 00011         CLRB    70(AP)                ; 1940
                                          50        01  D0 00014     MOVL    #1, R0                ; 1942
                                                    04  00017         RET                           ; 1944

; Routine Size:  24 bytes,    Routine Base:  _FOR$CODE + 0483

; 1898          1945  1 !<BLF/PAGE>
```

```
;  1900          1946  1 %SBTTL 'END_REPEAT - End a repeated value'
;  1901          1947  1 ROUTINE END_REPEAT =
;  1902          1948  1
;  1903          1949  1 !++
;  1904          1950  1 ! FUNCTIONAL DESCRIPTION:
;  1905          1951  1 !
;  1906          1952  1 !       LIB$TPARSE action routine which marks the end of a repeated value.
;  1907          1953  1 !
;  1908          1954  1 ! CALLING SEQUENCE:
;  1909          1955  1 !
;  1910          1956  1 !       status = END_REPEAT ()
;  1911          1957  1 !
;  1912          1958  1 ! FORMAL PARAMETERS:
;  1913          1959  1 !
;  1914          1960  1 !       NONE
;  1915          1961  1 !
;  1916          1962  1 ! IMPLICIT INPUTS:
;  1917          1963  1 !
;  1918          1964  1 !       AP      Points to PARAM_BLOCK
;  1919          1965  1 !
;  1920          1966  1 ! IMPLICIT OUTPUTS:
;  1921          1967  1 !
;  1922          1968  1 !       NML$T_TOKEN = 0, meaning that this value can't be an identifier
;  1923          1969  1 !       TPA$V_BLANKS = 0, disabling explicit blank processing
;  1924          1970  1 !
;  1925          1971  1 ! COMPLETION STATUS:
;  1926          1972  1 !
;  1927          1973  1 !       1 for success
;  1928          1974  1 !
;  1929          1975  1 ! SIDE EFFECTS:
;  1930          1976  1 !
;  1931          1977  1 !       NONE
;  1932          1978  1 !
;  1933          1979  1 !--
;  1934          1980  1
;  1935          1981  2     BEGIN
;  1936          1982  2
;  1937          1983  2     BUILTIN
;  1938          1984  2         AP;                 ! Argument pointer points to parameter block
;  1939          1985  2
;  1940          1986  2     MAP
;  1941          1987  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1942          1988  2
;  1943          1989  2     AP [NML$T_TOKEN] = 0;       ! Inhibit use of this token as an identifier
;  1944          1990  2     AP [TPA$V_BLANKS] = 0;      ! Turn off explicit blank processing
;  1945          1991  2
;  1946          1992  2     RETURN 1;
;  1947          1993  2
;  1948          1994  1     END;


                                0000 00000 END_REPEAT:
                                                .WORD   Save nothing                    ; 1947
                                7C   AC  94 00002       CLRB    124(AP)                 ; 1989
```

```
                            04    AC              01  8A 00005          BICB2   #1, 4(AP)                               ; 1990
                                  50              01  D0 00009          MOVL    #1, R0                                  ; 1992
                                                      04 0000C          RET                                             ; 1994

; Routine Size:  13 bytes,    Routine Base:  _FOR$CODE + 049B


; 1949          1995  1 !<BLF/PAGE>
```

```
;  1951          1996  1 %SBTTL 'STORE_VALUE - Store a value in a variable'
;  1952          1997  1 ROUTINE STORE_VALUE=
;  1953          1998  1
;  1954          1999  1 !++
;  1955          2000  1 ! FUNCTIONAL DESCRIPTION:
;  1956          2001  1 !
;  1957          2002  1 !       LIB$TPARSE action routine which stores the value just read in the
;  1958          2003  1 !       current variable.  If the repeat count is greater than 1, multiple
;  1959          2004  1 !       copies are moved.  However, if the value was of type CHARACTER,
;  1960          2005  1 !       all copies have been stored and this routine only returns success.
;  1961          2006  1 !       If the constant type is NULL, then "repeat-count" values are skipped.
;  1962          2007  1 !
;  1963          2008  1 ! CALLING SEQUENCE:
;  1964          2009  1 !
;  1965          2010  1 !       status = STORE_VALUE ()
;  1966          2011  1 !
;  1967          2012  1 ! FORMAL PARAMETERS:
;  1968          2013  1 !
;  1969          2014  1 !       NONE
;  1970          2015  1 !
;  1971          2016  1 ! IMPLICIT INPUTS:
;  1972          2017  1 !
;  1973          2018  1 !       AP      Points to PARAM_BLOCK
;  1974          2019  1 !
;  1975          2020  1 ! IMPLICIT OUTPUTS:
;  1976          2021  1 !
;  1977          2022  1 !       The user variable is modified (if value not NULL)
;  1978          2023  1 !
;  1979          2024  1 ! COMPLETION STATUS:
;  1980          2025  1 !
;  1981          2026  1 !       1 for success
;  1982          2027  1 !
;  1983          2028  1 ! SIDE EFFECTS:
;  1984          2029  1 !
;  1985          2030  1 !       Signals FOR$_SYNERRNAM if an error occurs during conversion.
;  1986          2031  1 !
;  1987          2032  1 !--
;  1988          2033  1
;  1989          2034  2     BEGIN
;  1990          2035  2
;  1991          2036  2     BUILTIN
;  1992          2037  2         AP;                 ! Argument pointer points to parameter block
;  1993          2038  2
;  1994          2039  2     MAP
;  1995          2040  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  1996          2041  2
;  1997          2042  2     !+
;  1998          2043  2     ! If this was a character string, all values have been stored.
;  1999          2044  2     !-
;  2000          2045  2
;  2001          2046  2     IF .AP [NML$B_CONSTYPE] EQL K_CHARACTER
;  2002          2047  2     THEN
;  2003          2048  2         RETURN 1;
;  2004          2049  2
;  2005          2050  2     !+
;  2006          2051  2     ! Check to see if we are past the end of the variable or array
;  2007          2052  2     !-
```

```
;  2008          2053  2
;  2009          2054  2      IF .AP [NML$A_VARSTART] GEQA .AP [NML$A_VAREND]
;  2010          2055  2      THEN
;  2011          2056  3          BEGIN
;  2012          2057  3          FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
;  2013          2058  3          RETURN 0;
;  2014          2059  2          END;
;  2015          2060  2
;  2016          2061  2      !+
;  2017          2062  2      ! If this was a repeated null (n*), then skip over values.
;  2018          2063  2      !-
;  2019          2064  2
;  2020          2065  2      IF .AP [NML$B_CONSTYPE] EQL K_NULL
;  2021          2066  2      THEN
;  2022          2067  2          WHILE .AP [NML$L_REPEATCT] GTR 0 DO
;  2023          2068  3              BEGIN
;  2024          2069  3              IF .AP [NML$A_VARCUR] GEQA .AP [NML$A_VAREND]
;  2025          2070  3              THEN
;  2026          2071  4                  BEGIN
;  2027          2072  4                  FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
;  2028          2073  4                  RETURN 0;
;  2029          2074  3                  END;
;  2030          2075  3              AP [NML$A_VARCUR] = .AP [NML$A_VARCUR] + .AP [NML$W_VARSIZE];
;  2031          2076  3              AP [NML$L_REPEATCT] = .AP [NML$L_REPEATCT] - 1;
;  2032          2077  3              END
;  2033          2078  2      ELSE
;  2034          2079  3          BEGIN
;  2035          2080  3
;  2036          2081  3          !+
;  2037          2082  3          ! Call routine to convert value to the appropriate destination type.
;  2038          2083  3          ! If conversion fails, signal an error.
;  2039          2084  3          !-
;  2040          2085  3
;  2041          2086  3          IF NOT FOR$$CVT_TYPE (.AP [NML$B_CONSTYPE], AP [NML$L_CONSBLOCK],
;  2042          2087  3                                .AP [NML$B_DTYPE], .AP [NML$A_VARSTART], 0)
;  2043          2088  3          THEN
;  2044          2089  3              CALLG (.AP, INPCONERR_ERROR);
;  2045          2090  3          AP [NML$A_VARCUR] = .AP [NML$A_VARSTART] + .AP [NML$W_VARSIZE];
;  2046          2091  3
;  2047          2092  3          !+
;  2048          2093  3          ! While repeat count is greater than 1, store copies of the value.
;  2049          2094  3          !-
;  2050          2095  3
;  2051          2096  3          WHILE .AP [NML$L_REPEATCT] GTR 1 DO
;  2052          2097  4              BEGIN
;  2053          2098  4              IF .AP [NML$A_VARCUR] GEQA .AP [NML$A_VAREND]
;  2054          2099  4              THEN
;  2055          2100  5                  BEGIN
;  2056          2101  5                  FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
;  2057          2102  5                  RETURN 0;
;  2058          2103  4                  END;
;  2059          2104  4              AP [NML$A_VARCUR] = CH$MOVE (.AP [NML$W_VARSIZE], .AP [NML$A_VARSTART],
;  2060          2105  4                                          .AP [NML$A_VARCUR]);
;  2061          2106  4              AP [NML$L_REPEATCT] = .AP [NML$L_REPEATCT] - 1;
;  2062          2107  3              END;
;  2063          2108  2          END;
;  2064          2109  2
```

```
2       !+
2       ! Turn off NML$V_IMAG if set.  This lets subsequent complex values get
2       ! stored correctly.
2       !-
2
2       AP [NML$V_IMAG] = 0;
2
2       !+
2       ! Update VARSTART with new position
2       !-
2
2       AP [NML$A_VARSTART] = .AP [NML$A_VARCUR];
2       RETURN 1;
1       END;
```

```
                                   00FC 00000 STORE_VALUE:
                                              .WORD   Save R2,R3,R4,R5,R6,R7          ; 1997
             05       46   AC   91 00002      CMPB    70(AP), #5                      ; 2046
                           03   12 00006      BNEQ    1$
                         0080   31 00008      BRW     9$
        30   AC       2C   AC   D1 0000B 1$:  CMPL    44(AP), 48(AP)                  ; 2054
                           55   1E 00010      BGEQU   6$
             57       78   AC   9E 00012      MOVAB   120(AP), R7                     ; 2067
             56       34   AC   9E 00016      MOVAB   52(AP), R6                      ; 2069
                      46   AC   95 0001A      TSTB    70(AP)                          ; 2065
                           15   12 0001D      BNEQ    3$
                           67   D5 0001F 2$:  TSTL    (R7)                            ; 2067
                           60   15 00021      BLEQ    8$
        30   AC            66   D1 00023      CMPL    (R6), 48(AP)                    ; 2069
                           3E   1E 00027      BGEQU   6$
             50       38   AC   3C 00029      MOVZWL  56(AP), R0                      ; 2075
             66            50   C0 0002D      ADDL2   R0, (R6)
                           67   D7 00030      DECL    (R7)                            ; 2076
                           EB   11 00032      BRB     2$                              ; 2067
                           7E   D4 00034 3$:  CLRL    -(SP)                           ; 2086
                      2C   AC   DD 00036      PUSHL   44(AP)                          ; 2087
             7E       44   AC   9A 00039      MOVZBL  68(AP), -(SP)
                      68   AC   9F 0003D      PUSHAB  104(AP)                         ; 2086
             7E       46   AC   9A 00040      MOVZBL  70(AP), -(SP)
  00000000G  00            05   FB 00044      CALLS   #5, FOR$$CVT_TYPE
             05            50   E8 0004B      BLBS    R0, 4$
      0000V  CF            6C   FA 0004E      CALLG   (AP), INPCONERR_ERROR           ; 2089
             50       38   AC   3C 00053 4$:  MOVZWL  56(AP), R0                      ; 2090
             66       2C BC40   9E 00057      MOVAB   @44(AP)[R0], (R6)
             01            67   D1 0005C 5$:  CMPL    (R7), #1                        ; 2096
                           22   15 0005F      BLEQ    8$
        30   AC            66   D1 00061      CMPL    (R6), 48(AP)                    ; 2098
                           0E   1F 00065      BLSSU   7$
                      28   AC   DD 00067 6$:  PUSHL   40(AP)                          ; 2101
                           12   DD 0006A      PUSHL   #18
  00000000G  00            02   FB 0006C      CALLS   #2, FOR$$SIGNAL_STO
                           1A   11 00073      BRB     10$                             ; 2102
 00  B6  2C  BC       38   AC   28 00075 7$:  MOVC3   56(AP), @44(AP), @0(R6)         ; 2105
```

```
                  66         53  D0 0007C          MOVL    R3, (R6)              ;
                             67  D7 0007F          DECL    (R7)                  ; 2106
                             D9  11 00081          BRB     5$                    ; 2096
            45    AC         02  8A 00083 8$:      BICB2   #2, 69(AP)            ; 2115
            2C    AC         66  D0 00087          MOVL    (R6), 44(AP)          ; 2121
                  50         01  D0 0008B 9$:      MOVL    #1, R0                ; 2122
                                 04 0008E          RET                           ;
                             50  D4 0008F 10$:     CLRL    R0                    ; 2123
                                 04 00091          RET                           ;

; Routine Size:  146 bytes,    Routine Base:  _FOR$CODE + 04A8

; 2079          2124  1 !<BLF/PAGE>
```

```
;  2081          2125  1 %SBTTL 'NULL_VALUE - Skip an element'
;  2082          2126  1 ROUTINE NULL_VALUE =
;  2083          2127  1
;  2084          2128  1 !++
;  2085          2129  1 ! FUNCTIONAL DESCRIPTION:
;  2086          2130  1 !
;  2087          2131  1 !       LIBSTPARSE action routine which is called when a comma is found in place
;  2088          2132  1 !       of a value.  The pointer to the current element is advanced one element
;  2089          2133  1 !       with no change being made to the current element.  Note that if the
;  2090          2134  1 !       current variable is not an array, an attempt to store a following value
;  2091          2135  1 !       will be an error.  If we have already passed the last element, give
;  2092          2136  1 !       an error.
;  2093          2137  1 !
;  2094          2138  1 ! CALLING SEQUENCE:
;  2095          2139  1 !
;  2096          2140  1 !       status = NULL_VALUE ()
;  2097          2141  1 !
;  2098          2142  1 ! FORMAL PARAMETERS:
;  2099          2143  1 !
;  2100          2144  1 !       NONE
;  2101          2145  1 !
;  2102          2146  1 ! IMPLICIT INPUTS:
;  2103          2147  1 !
;  2104          2148  1 !       AP      Points to PARAM_BLOCK
;  2105          2149  1 !
;  2106          2150  1 ! IMPLICIT OUTPUTS:
;  2107          2151  1 !
;  2108          2152  1 !       NML$A_VARSTART is advanced one element.
;  2109          2153  1 !       NML$A_VARCUR = NML$A_VARSTART
;  2110          2154  1 !
;  2111          2155  1 ! COMPLETION STATUS:
;  2112          2156  1 !
;  2113          2157  1 !       1
;  2114          2158  1 !
;  2115          2159  1 ! SIDE EFFECTS:
;  2116          2160  1 !
;  2117          2161  1 !       FOR$_TOOMANVAL - if this comma implies a skip past the end of the
;  2118          2162  1 !       variable.
;  2119          2163  1 !
;  2120          2164  1 !--
;  2121          2165  1
;  2122          2166  2     BEGIN
;  2123          2167  2
;  2124          2168  2     BUILTIN
;  2125          2169  2         AP;                 ! Argument pointer points to parameter block
;  2126          2170  2
;  2127          2171  2     MAP
;  2128          2172  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  2129          2173  2
;  2130          2174  2     !+
;  2131          2175  2     ! If we are already past the end of the variable, give an error.
;  2132          2176  2     !-
;  2133          2177  2
;  2134          2178  2     IF .AP [NML$A_VARSTART] GEQA .AP [NML$A_VAREND]
;  2135          2179  2     THEN
;  2136          2180  2         FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
;  2137          2181  2
```

```
; 2138          2182  2        IF .AP [NML$W_STRIDE] NEQ 0
; 2139          2183  2        THEN
; 2140          2184  2            AP [NML$A_VARSTART] = .AP [NML$A_VARSTART] + .AP [NML$W_STRIDE]
; 2141          2185  2        ELSE
; 2142          2186  2            AP [NML$A_VARSTART] = .AP [NML$A_VAREND];
; 2143          2187  2
; 2144          2188  2        AP [NML$A_VARCUR] = .AP [NML$A_VARSTART];
; 2145          2189  2
; 2146          2190  2        RETURN 1;
; 2147          2191  2
; 2148          2192  1        END;


                                          0000 00000 NULL_VALUE:
                                                             .WORD   Save nothing                                    ; 2126
                  30   AC       2C   AC   D1 00002           CMPL    44(AP), 48(AP)                                  ; 2178
                                     0C   1F 00007           BLSSU   1$                                              ;
                                28   AC   DD 00009           PUSHL   40(AP)                                          ; 2180
                                     12   DD 0000C           PUSHL   #18                                             ;
           00000000G   00            02   FB 0000E           CALLS   #2, FOR$$SIGNAL_STO                             ;
                                3A   AC   B5 00015 1$:       TSTW    58(AP)                                          ; 2182
                                     0A   13 00018           BEQL    2$                                              ;
                       50       3A   AC   3C 0001A           MOVZWL  58(AP), R0                                      ; 2184
                  2C   AC            50   C0 0001E           ADDL2   R0, 44(AP)                                      ;
                                     05   11 00022           BRB     3$                                              ;
                  2C   AC       30   AC   D0 00024 2$:       MOVL    48(AP), 44(AP)                                  ; 2186
                  34   AC       2C   AC   D0 00029 3$:       MOVL    44(AP), 52(AP)                                  ; 2188
                       50            01   D0 0002E           MOVL    #1, R0                                          ; 2190
                                          04 00031           RET                                                     ; 2192

; Routine Size:  50 bytes,    Routine Base:  _FOR$CODE + 053A


; 2149          2193  1 !<BLF/PAGE>
```

```
;  2151          2194  1 %SBTTL 'SYNTAX_ERROR - Signal syntax error'
;  2152          2195  1 ROUTINE SYNTAX_ERROR =
;  2153          2196  1
;  2154          2197  1 !++
;  2155          2198  1 ! FUNCTIONAL DESCRIPTION:
;  2156          2199  1 !
;  2157          2200  1 !       LIB$TPARSE action routine which signals a syntax error.
;  2158          2201  1 !
;  2159          2202  1 ! CALLING SEQUENCE:
;  2160          2203  1 !
;  2161          2204  1 !       status = SYNTAX_ERROR ()
;  2162          2205  1 !
;  2163          2206  1 ! FORMAL PARAMETERS:
;  2164          2207  1 !
;  2165          2208  1 !       NONE
;  2166          2209  1 !
;  2167          2210  1 ! IMPLICIT INPUTS:
;  2168          2211  1 !
;  2169          2212  1 !       AP      Points to PARAM_BLOCK
;  2170          2213  1 !
;  2171          2214  1 ! IMPLICIT OUTPUTS:
;  2172          2215  1 !
;  2173          2216  1 !       NONE
;  2174          2217  1 !
;  2175          2218  1 ! COMPLETION STATUS:
;  2176          2219  1 !
;  2177          2220  1 !       NONE
;  2178          2221  1 !
;  2179          2222  1 ! SIDE EFFECTS:
;  2180          2223  1 !
;  2181          2224  1 !       Signals FOR$_SYNERRNAM - Syntax error in NAMELIST
;  2182          2225  1 !
;  2183          2226  1 !--
;  2184          2227  1
;  2185          2228  2     BEGIN
;  2186          2229  2
;  2187          2230  2     BUILTIN
;  2188          2231  2         AP;                 ! Argument pointer points to parameter block
;  2189          2232  2
;  2190          2233  2     MAP
;  2191          2234  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  2192          2235  2
;  2193          2236  2     IF .AP [TPA$L_TOKENCNT] LSS 6
;  2194          2237  2     THEN
;  2195          2238  3         BEGIN
;  2196          2239  3         LOCAL
;  2197          2240  3             EXTRA,
;  2198          2241  3             CCB: REF $FOR$CCB_DECL;
;  2199          2242  3         CCB = .AP [NML$A_CCB];
;  2200          2243  3
;  2201          2244  3         !+
;  2202          2245  3         ! Try to make the string reported include the part of the record
;  2203          2246  3         ! where the error was.
;  2204          2247  3         !-
;  2205          2248  3
;  2206          2249  3         IF .AP [TPA$L_TOKENPTR] GEQA .CCB [LUB$A_BUF_PTR]
;  2207          2250  3         THEN
```

```
; 2208          2251  4                 BEGIN
; 2209          2252  4                 EXTRA = MAX (0, (6 - .AP [TPA$L_TOKENCNT]));
; 2210          2253  4                 IF .AP [TPA$L_TOKENPTR] - .EXTRA LSSA .CCB [LUB$A_BUF_PTR]
; 2211          2254  4                 THEN
; 2212          2255  4                     EXTRA = .AP [TPA$L_TOKENPTR] - .CCB [LUB$A_BUF_PTR];
; 2213          2256  4                 AP [TPA$L_TOKENCNT] = .AP [TPA$L_TOKENCNT] + .EXTRA;
; 2214          2257  4                 AP [TPA$L_TOKENPTR] = .AP [TPA$L_TOKENPTR] - .EXTRA;
; 2215          2258  3                 END;
; 2216          2259  2             END;
; 2217          2260  2
; 2218          2261  2         FOR$$SIGNAL_STO (FOR$K_SYNERRNAM, AP [TPA$L_TOKENCNT]);
; 2219          2262  2         RETURN 0;
; 2220          2263  2
; 2221          2264  1         END;



                                        0004 00000 SYNTAX_ERROR:
                                                          .WORD   Save R2                                  ; 2195
                       06       10  AC  D1 00002          CMPL    16(AP), #6                               ; 2236
                                    30  18 00006          BGEQ    3$                                       ;
                       51       40  AC  D0 00008          MOVL    64(AP), CCB                              ; 2242
                  B0   A1       14  AC  D1 0000C          CMPL    20(AP), -80(CCB)                         ; 2249
                                    25  1F 00011          BLSSU   3$                                       ;
          50           06       10  AC  C3 00013          SUBL3   16(AP), #6, R0                           ; 2252
                                    02  18 00018          BGEQ    1$                                       ;
                                    50  D4 0001A          CLRL    R0                                       ;
                       52           50  D0 0001C 1$:      MOVL    R0, EXTRA                                ;
          50      14   AC           52  C3 0001F          SUBL3   EXTRA, 20(AP), R0                        ; 2253
                  B0   A1           50  D1 00024          CMPL    R0, -80(CCB)                             ;
                                    06  1E 00028          BGEQU   2$                                       ;
          52      14   AC       B0  A1  C3 0002A          SUBL3   -80(CCB), 20(AP), EXTRA                  ; 2255
                  10   AC           52  C0 00030 2$:      ADDL2   EXTRA, 16(AP)                            ; 2256
                  14   AC           52  C2 00034          SUBL2   EXTRA, 20(AP)                            ; 2257
                                10  AC  9F 00038 3$:      PUSHAB  16(AP)                                   ; 2261
                                    11  DD 0003B          PUSHL   #17                                      ;
             00000000G 00           02  FB 0003D          CALLS   #2, FOR$$SIGNAL_STO                      ;
                                    50  D4 00044          CLRL    R0                                       ; 2262
                                        04 00046          RET                                              ; 2264

; Routine Size:  71 bytes,    Routine Base:  _FOR$CODE + 056C


; 2222          2265  1 !<BLF/PAGE>
```

```
;  2224          2266  1 %SBTTL 'INVREFVAR_ERROR - Signal invalid variable reference error'
;  2225          2267  1 ROUTINE INVREFVAR_ERROR =
;  2226          2268  1
;  2227          2269  1 !++
;  2228          2270  1 ! FUNCTIONAL DESCRIPTION:
;  2229          2271  1 !
;  2230          2272  1 !       LIBSTPARSE action routine which signals an invalid variable
;  2231          2273  1 !       reference  error.
;  2232          2274  1 !
;  2233          2275  1 ! CALLING SEQUENCE:
;  2234          2276  1 !
;  2235          2277  1 !       status = INVREFVAR_ERROR ()
;  2236          2278  1 !
;  2237          2279  1 ! FORMAL PARAMETERS:
;  2238          2280  1 !
;  2239          2281  1 !       NONE
;  2240          2282  1 !
;  2241          2283  1 ! IMPLICIT INPUTS:
;  2242          2284  1 !
;  2243          2285  1 !       AP      Points to PARAM_BLOCK
;  2244          2286  1 !
;  2245          2287  1 ! IMPLICIT OUTPUTS:
;  2246          2288  1 !
;  2247          2289  1 !       NONE
;  2248          2290  1 !
;  2249          2291  1 ! COMPLETION STATUS:
;  2250          2292  1 !
;  2251          2293  1 !       NONE
;  2252          2294  1 !
;  2253          2295  1 ! SIDE EFFECTS:
;  2254          2296  1 !
;  2255          2297  1 !       Signals FOR$_INVREFVAR - Invalid reference to variable in NAMELIST
;  2256          2298  1 !
;  2257          2299  1 !--
;  2258          2300  1
;  2259          2301  2     BEGIN
;  2260          2302  2
;  2261          2303  2     BUILTIN
;  2262          2304  2         AP;                 ! Argument pointer points to parameter block
;  2263          2305  2
;  2264          2306  2     MAP
;  2265          2307  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  2266          2308  2
;  2267          2309  2     LOCAL
;  2268          2310  2         DESCR: VECTOR [2, LONG],
;  2269          2311  2         VARNAME: REF VECTOR [, BYTE];
;  2270          2312  2
;  2271          2313  2     VARNAME = .AP [NML$A_VARNAME];
;  2272          2314  2     DESCR [0] = .VARNAME [0];
;  2273          2315  2     DESCR [1] = VARNAME [1];
;  2274          2316  2     FOR$$SIGNAL_STO (FOR$K_INVREFVAR, DESCR);
;  2275          2317  2     RETURN 0;
;  2276          2318  2
;  2277          2319  1     END;
```

```
                                          0000 00000 INVREFVAR_ERROR:
                                                              .WORD   Save nothing                           ; 2267
                     5E             04  C2 00002              SUBL2   #4, SP                                 ;
                     50       28    AC  D0 00005              MOVL    40(AP), VARNAME                        ; 2313
                     7E             60  9A 00009              MOVZBL  (VARNAME), DESCR                       ; 2314
              04     AE       01    A0  9E 0000C              MOVAB   1(R0), DESCR+4                         ; 2315
                                    5E  DD 00011              PUSHL   SP                                     ; 2316
                                    13  DD 00013              PUSHL   #19                                    ;
           00000000G 00             02  FB 00015              CALLS   #2, FOR$$SIGNAL_STO                    ;
                                    50  D4 0001C              CLRL    R0                                     ; 2317
                                        04 0001E              RET                                            ; 2319

; Routine Size:  31 bytes,    Routine Base:  _FOR$CODE + 05B3

; 2278          2320  1 !<BLF/PAGE>
```

```
;  2280          2321  1 %SBTTL 'INPCONERR_ERROR - Signal input conversion error'
;  2281          2322  1 ROUTINE INPCONERR_ERROR =
;  2282          2323  1
;  2283          2324  1 !++
;  2284          2325  1 ! FUNCTIONAL DESCRIPTION:
;  2285          2326  1 !
;  2286          2327  1 !       Routine which signals FOR$_INPCONERR, "input conversion error",
;  2287          2328  1 !       with a chained message giving the text and record number.  Although
;  2288          2329  1 !       called as if it were a LIB$TPARSE action routine, in fact it is
;  2289          2330  1 !       only called from other action routines.
;  2290          2331  1 !
;  2291          2332  1 ! CALLING SEQUENCE:
;  2292          2333  1 !
;  2293          2334  1 !       status = INPCONERR_ERROR ()
;  2294          2335  1 !
;  2295          2336  1 ! FORMAL PARAMETERS:
;  2296          2337  1 !
;  2297          2338  1 !       NONE
;  2298          2339  1 !
;  2299          2340  1 ! IMPLICIT INPUTS:
;  2300          2341  1 !
;  2301          2342  1 !       AP      Points to PARAM_BLOCK
;  2302          2343  1 !
;  2303          2344  1 ! IMPLICIT OUTPUTS:
;  2304          2345  1 !
;  2305          2346  1 !       NONE
;  2306          2347  1 !
;  2307          2348  1 ! COMPLETION STATUS:
;  2308          2349  1 !
;  2309          2350  1 !       NONE
;  2310          2351  1 !
;  2311          2352  1 ! SIDE EFFECTS:
;  2312          2353  1 !
;  2313          2354  1 !       Signals FOR$_INPCONERR, input conversion error
;  2314          2355  1 !
;  2315          2356  1 !--
;  2316          2357  1
;  2317          2358  2     BEGIN
;  2318          2359  2
;  2319          2360  2     BUILTIN
;  2320          2361  2         AP;                 ! Argument pointer points to parameter block
;  2321          2362  2
;  2322          2363  2     MAP
;  2323          2364  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  2324          2365  2
;  2325          2366  2     LOCAL
;  2326          2367  2         CCB: REF $FOR$CCB_DECL;
;  2327          2368  2
;  2328          2369  2     CCB = .AP [NML$A_CCB];      ! Get CCB address
;  2329          2370  2
;  2330          2371  2     !+
;  2331          2372  2     ! If the file is indexed organization or is an internal file (unlikely,
;  2332          2373  2     ! since that's not allowed), then use the message that doesn't have
;  2333          2374  2     ! a record number.  Otherwise chain the message with both text and
;  2334          2375  2     ! record number.  Signal it as a continuable error.
;  2335          2376  2     !-
;  2336          2377  2
```

```
; 2337          2378  2      IF (.CCB [LUB$B_ORGAN] EQL LUB$K_ORG_INDEX) OR
; 2338          2379  3         (.CCB [LUB$W_LUN] EQL LUB$K_LUN_ENCD)
; 2339          2380  2      THEN
; 2340          2381  2         FOR$$SIGNAL (FOR$K_INPCONERR, FOR$_INVTEX, 1, AP [TPA$L_TOKENCNT])
; 2341          2382  2      ELSE
; 2342          2383  2         FOR$$SIGNAL (FOR$K_INPCONERR, FOR$_INVTEXREC, 2,
; 2343          2384  2            AP [TPA$L_TOKENCNT], .CCB [LUB$L_LOG_RECNO] - 1);
; 2344          2385  2      RETURN 0;
; 2345          2386  2
; 2346          2387  1      END;



                                        001C 00000 INPCONERR_ERROR:
                                                          .WORD   Save R2,R3,R4                     ; 2322
                     54 0000000G  00   9E 00002           MOVAB   FOR$$SIGNAL, R4
                     52       40  AC   D0 00009           MOVL    64(AP), CCB                       ; 2369
                     53       10  AC   9E 0000D           MOVAB   16(AP), R3                        ; 2381
                     03       C4  A2   91 00011           CMPB    -60(CCB), #3                      ; 2378
                                  08   13 00015           BEQL    1$
          FFFB       8F       C6  A2   B1 00017           CMPW    -58(CCB), #-5                     ; 2379
                                  13   12 0001D           BNEQ    2$
                                  53   DD 0001F 1$:       PUSHL   R3                                ; 2381
                                  01   DD 00021           PUSHL   #1
                       0018883C   8F   DD 00023           PUSHL   #1607740
                     7E       40  8F   9A 00029           MOVZBL  #64, -(SP)
                     64           04   FB 0002D           CALLS   #4, FOR$$SIGNAL
                                  16   11 00030           BRB     3$
     7E       E0     A2           01   C3 00032 2$:       SUBL3   #1, -32(CCB), -(SP)               ; 2384
                                  53   DD 00037           PUSHL   R3
                                  02   DD 00039           PUSHL   #2
                       00188834   8F   DD 0003B           PUSHL   #1607732
                     7E       40  8F   9A 00041           MOVZBL  #64, -(SP)
                     64           05   FB 00045           CALLS   #5, FOR$$SIGNAL
                                  50   D4 00048 3$:       CLRL    R0                                ; 2385
                                       04 0004A           RET                                       ; 2387

; Routine Size:  75 bytes,    Routine Base:  _FOR$CODE + 05D2


; 2347          2388  1 !<BLF/PAGE>
```

```
%SBTTL 'BLANKS_OFF - Turn off explicit blanks'
ROUTINE BLANKS_OFF =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Turns off explicit blank processing for LIB$TPARSE.  When off, blanks
!       are implicit separators.
!
! CALLING SEQUENCE:
!
!       status = BLANKS_OFF ()
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       AP      Points to PARAM_BLOCK
!
! IMPLICIT OUTPUTS:
!
!       PARAM_BLOCK [TPA$V_BLANKS] = 0
!
! COMPLETION STATUS:
!
!       1 for success
!
! SIDE EFFECTS:
!
!       NONE
!
!--

   BEGIN

   BUILTIN
       AP;              ! Argument pointer points to parameter block

   MAP
       AP: REF BLOCK [, BYTE] FIELD (NMLSFIELDS);

   AP [TPA$V_BLANKS] = 0;       ! Turn off blank processing
   RETURN 1;

   END;
```

```
                              0000 00000 BLANKS_OFF:
                                               .WORD   Save nothing          ; 2390
        04   AC            01   8A 00002       BICB2   #1, 4(AP)             ; 2432
             50            01   D0 00006       MOVL    #1, R0                ; 2433
                                04 00009       RET                           ; 2435
```

; Routine Size: 10 bytes,   Routine Base: _FOR$CODE + 061D

; 2396   2436 1 !<BLF/PAGE>

```
; 2398      2437  1 %SBTTL 'BLANKS_ON - Turn on explicit blanks'
; 2399      2438  1 ROUTINE BLANKS_ON =
; 2400      2439  1
; 2401      2440  1 !++
; 2402      2441  1 ! FUNCTIONAL DESCRIPTION:
; 2403      2442  1 !
; 2404      2443  1 !       Turns on explicit blank processing for LIB$TPARSE.  When on, blanks
; 2405      2444  1 !       are not implicit separators.
; 2406      2445  1 !
; 2407      2446  1 ! CALLING SEQUENCE:
; 2408      2447  1 !
; 2409      2448  1 !       status = BLANKS_ON ()
; 2410      2449  1 !
; 2411      2450  1 ! FORMAL PARAMETERS:
; 2412      2451  1 !
; 2413      2452  1 !       NONE
; 2414      2453  1 !
; 2415      2454  1 ! IMPLICIT INPUTS:
; 2416      2455  1 !
; 2417      2456  1 !       AP      Points to PARAM_BLOCK
; 2418      2457  1 !
; 2419      2458  1 ! IMPLICIT OUTPUTS:
; 2420      2459  1 !
; 2421      2460  1 !       PARAM_BLOCK [TPA$V_BLANKS] = 0
; 2422      2461  1 !
; 2423      2462  1 ! COMPLETION STATUS:
; 2424      2463  1 !
; 2425      2464  1 !       1 for success
; 2426      2465  1 !
; 2427      2466  1 ! SIDE EFFECTS:
; 2428      2467  1 !
; 2429      2468  1 !       NONE
; 2430      2469  1 !
; 2431      2470  1 !--
; 2432      2471  1
; 2433      2472  2     BEGIN
; 2434      2473  2
; 2435      2474  2     BUILTIN
; 2436      2475  2         AP;                 ! Argument pointer points to parameter block
; 2437      2476  2
; 2438      2477  2     MAP
; 2439      2478  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
; 2440      2479  2
; 2441      2480  2     AP [TPA$V_BLANKS] = 1;      ! Turn on blank processing
; 2442      2481  2     RETURN 1;
; 2443      2482  2
; 2444      2483  1     END;



                                 0000 00000 BLANKS_ON:
                                                   .WORD   Save nothing                    ; 2438
        04   AC            01  88 00002            BISB2   #1, 4(AP)                       ; 2480
             50            01  D0 00006            MOVL    #1, R0                          ; 2481
                               04 00009            RET                                     ; 2483
```

; Routine Size: 10 bytes,   Routine Base: _FOR$CODE + 0627

; 2445 2484 1 !<BLF/PAGE>

```
;  2447          2485  1 %SBTTL 'LOOKUP_IDENTIFIER - Lookup identifier in NAMELIST group'
;  2448          2486  1 ROUTINE LOOKUP_IDENTIFIER =
;  2449          2487  1
;  2450          2488  1 !++
;  2451          2489  1 ! FUNCTIONAL DESCRIPTION:
;  2452          2490  1 !
;  2453          2491  1 !       Searches the NAMELIST group for an identifier which matches the
;  2454          2492  1 !       current token.  If found, the descriptor information is entered into
;  2455          2493  1 !       the parameter block.  If not found, an error is signalled.
;  2456          2494  1 !
;  2457          2495  1 ! CALLING SEQUENCE:
;  2458          2496  1 !
;  2459          2497  1 !       status = LOOKUP_IDENTIFIER ()
;  2460          2498  1 !
;  2461          2499  1 ! FORMAL PARAMETERS:
;  2462          2500  1 !
;  2463          2501  1 !       NONE
;  2464          2502  1 !
;  2465          2503  1 ! IMPLICIT INPUTS:
;  2466          2504  1 !
;  2467          2505  1 !       AP      Points to PARAM_BLOCK
;  2468          2506  1 !
;  2469          2507  1 ! IMPLICIT OUTPUTS:
;  2470          2508  1 !
;  2471          2509  1 !       PARAM_BLOCK [NML$A_VARNAME] = address of variable name counted string
;  2472          2510  1 !       PARAM_BLOCK [NML$A_VARSTART] = address of variable low byte
;  2473          2511  1 !       PARAM_BLOCK [NML$A_VAREND] = address of next byte past end of variable
;  2474          2512  1 !       PARAM_BLOCK [NML$A_VARCUR] = same as VARSTART
;  2475          2513  1 !       PARAM_BLOCK [NML$W_VARSIZE] = size of a variable element in bytes
;  2476          2514  1 !       PARAM_BLOCK [NML$W_STRIDE] = stride between elements if array, else 0
;  2477          2515  1 !       PARAM_BLOCK [NML$B_DTYPE] = descriptor datatype code of variable
;  2478          2516  1 !       PARAM_BLOCK [NML$B_CONSTYPE] = 0
;  2479          2517  1 !       PARAM_BLOCK [NML$L_REPEATCT] = 1
;  2480          2518  1 !       PARAM_BLOCK [NML$V_IMAG] = 0
;  2481          2519  1 !       PARAM_BLOCK [NML$V_VALUE_IDENT] = 0
;  2482          2520  1 !       PARAM_BLOCK [NML$V_SUBSTRING] = 0;
;  2483          2521  1 !       PARAM_BLOCK [NML$V_SUBSCRIPT] = 0;
;  2484          2522  1 !
;  2485          2523  1 ! COMPLETION STATUS:
;  2486          2524  1 !
;  2487          2525  1 !       1 for success
;  2488          2526  1 !
;  2489          2527  1 ! SIDE EFFECTS:
;  2490          2528  1 !
;  2491          2529  1 !       Signals FOR$_INVREFVAR - Invalid NAMELIST variable if identifier is not in
;  2492          2530  1 !       the current group.
;  2493          2531  1 !
;  2494          2532  1 !--
;  2495          2533  1
;  2496          2534  2     BEGIN
;  2497          2535  2
;  2498          2536  2     BUILTIN
;  2499          2537  2         AP;                 ! Argument pointer points to parameter block
;  2500          2538  2
;  2501          2539  2     MAP
;  2502          2540  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  2503          2541  2
```

```
;  2504          2542  2     LOCAL
;  2505          2543  2         NML_LIST: REF VECTOR [, LONG];          ! Pointer to list descriptor
;  2506          2544  2
;  2507          2545  2     NML_LIST = .AP [NML$A_LISTBLOCK];           ! Get list block address
;  2508          2546  2
;  2509          2547  2     !+
;  2510          2548  2     ! Loop through identifier list looking for a matching identifier.  If none
;  2511          2549  2     ! found, signal an error.  Loop value will be true if no match found.
;  2512          2550  2     !-
;  2513          2551  2
;  2514          2552  2     IF (
;  2515          2553  3         DECRU I FROM .(NML_LIST [1])<0,16,0> TO 1 DO    ! Count is first word of second longword
;  2516          2554  4             BEGIN
;  2517          2555  4             NML_LIST = NML_LIST [2];                ! Move to next identifier in list
;  2518          2556  4             IF COMPARE_UPCASE (.NML_LIST[0], AP [TPA$L_TOKENCNT])
;  2519          2557  4             THEN
;  2520          2558  4                 EXITLOOP 0;     ! Loop value false if a match is found
;  2521          2559  3             END)
;  2522          2560  2     THEN
;  2523          2561  3         BEGIN
;  2524          2562  3         !+
;  2525          2563  3         ! If we get here, there is no match.  Signal an error giving the variable
;  2526          2564  3         ! name.
;  2527          2565  3         !-
;  2528          2566  3
;  2529          2567  3         FOR$$SIGNAL_STO (FOR$K_INVREFVAR, AP [TPA$L_TOKENCNT]);
;  2530          2568  3
;  2531          2569  3         RETURN 0;       ! Execution should never return here
;  2532          2570  3         END
;  2533          2571  2
;  2534          2572  2     ELSE
;  2535          2573  2
;  2536          2574  3         BEGIN
;  2537          2575  3
;  2538          2576  3         !+
;  2539          2577  3         ! A match has been found.  Fill in the parameter block from the
;  2540          2578  3         ! descriptor.
;  2541          2579  3         !-
;  2542          2580  3
;  2543          2581  3         LOCAL
;  2544          2582  3             DESC: REF BLOCK [, BYTE];   ! Variable descriptor
;  2545          2583  3
;  2546          2584  3         AP [NML$A_VARNAME] = .NML_LIST [0];     ! Address of name counted string
;  2547          2585  3         DESC = .NML_LIST [1];               ! Descriptor address
;  2548          2586  3
;  2549          2587  3         !+
;  2550          2588  3         ! Validate descriptor class and datatype
;  2551          2589  3         ! \\ Note:  The use of the ONE_OF macro here assumes that
;  2552          2590  3         !    neither a datatype code of 0 nor a class code
;  2553          2591  3         !    of 0 is one of the valid ones.  If this
;  2554          2592  3         !    is no longer true, the value must first be tested to
;  2555          2593  3         !    ensure that it is not greater than 127 (unsigned). \\
;  2556          2594  3         !-
;  2557          2595  3
;  2558        P 2596  3         IF NOT ONE_OF (.DESC [DSC$B_DTYPE],
;  2559        P 2597  3             DSC$K_DTYPE_BU, DSC$K_DTYPE_B , DSC$K_DTYPE_WU,
;  2560        P 2598  3             DSC$K_DTYPE_W , DSC$K_DTYPE_LU, DSC$K_DTYPE_L ,
```

```
;  2561        P 2599  3              DSC$K_DTYPE_T , DSC$K_DTYPE_F , DSC$K_DTYPE_D ,
;  2562        P 2600  3              DSC$K_DTYPE_G , DSC$K_DTYPE_H , DSC$K_DTYPE_FC,
;  2563          2601  3              DSC$K_DTYPE_DC, DSC$K_DTYPE_GC) OR
;  2564        P 2602  3            NOT ONE_OF (.DESC [DSC$B_CLASS],
;  2565          2603  4              DSC$K_CLASS_S, DSC$K_CLASS_A)
;  2566          2604  3          THEN
;  2567          2605  4              BEGIN
;  2568          2606  4              FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
;  2569          2607  4              RETURN 0;
;  2570          2608  3              END;
;  2571          2609  3
;  2572          2610  3          !+
;  2573          2611  3          ! Fill in parameter block.
;  2574          2612  3          !-
;  2575          2613  3
;  2576          2614  3          AP [NML$A_VARSTART] = .DESC [DSC$A_POINTER];
;  2577          2615  3          AP [NML$A_VARCUR] = .DESC [DSC$A_POINTER];
;  2578          2616  3          AP [NML$W_VARSIZE] = .DESC [DSC$W_LENGTH];
;  2579          2617  3          AP [NML$B_DTYPE] = .DESC [DSC$B_DTYPE];
;  2580          2618  3          AP [NML$A_DESCR] = .DESC;
;  2581          2619  3
;  2582          2620  3          IF .DESC [DSC$B_CLASS] EQL DSC$K_CLASS_A
;  2583          2621  3          THEN
;  2584          2622  4              BEGIN
;  2585          2623  4              !+
;  2586          2624  4              ! If the array descriptor doesn't have COLUMN order and
;  2587          2625  4              ! coefficient and bounds blocks, or if it has
;  2588          2626  4              ! more than 7 dimensions, then the descriptor is
;  2589          2627  4              ! invalid for us.
;  2590          2628  4              !-
;  2591          2629  4
;  2592          2630  5              IF NOT (.DESC [DSC$V_FL_COLUMN] AND
;  2593          2631  5                      .DESC [DSC$V_FL_COEFF] AND
;  2594          2632  5                      .DESC [DSC$V_FL_BOUNDS] AND
;  2595          2633  5                      (.DESC [DSC$B_DIMCT] LEQU 7))
;  2596          2634  4              THEN
;  2597          2635  5                  BEGIN
;  2598          2636  5                  FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
;  2599          2637  5                  RETURN 0;
;  2600          2638  4                  END;
;  2601          2639  4
;  2602          2640  4              AP [NML$A_VAREND] = .AP [NML$A_VARSTART] + .DESC [DSC$L_ARSIZE];
;  2603          2641  4              AP [NML$W_STRIDE] = .AP [NML$W_VARSIZE];
;  2604          2642  4              END
;  2605          2643  3          ELSE
;  2606          2644  4              BEGIN
;  2607          2645  4              AP [NML$A_VAREND] = .AP [NML$A_VARSTART] + .AP [NML$W_VARSIZE];
;  2608          2646  4              AP [NML$W_STRIDE] = 0;
;  2609          2647  3              END;
;  2610          2648  3
;  2611          2649  3          AP [NML$B_CONSTYPE] = 0;
;  2612          2650  3          AP [NML$L_REPEATCT] = 1;
;  2613          2651  3          AP [NML$V_IMAG] = 0;
;  2614          2652  3          AP [NML$V_VALUE_IDENT] = 0;
;  2615          2653  3          AP [NML$V_SUBSCRIPT] = 0;
;  2616          2654  3          AP [NML$V_SUBSTRING] = 0;
;  2617          2655  3
```

```
; 2618        2656  3          !+
; 2619        2657  3          ! Since FORTRAN insists on passing us datatype BU for a signed byte,
; 2620        2658  3          ! change it here.
; 2621        2659  3          !-
; 2622        2660  3
; 2623        2661  3          IF .AP [NML$B_DTYPE] EQL DSC$K_DTYPE_BU
; 2624        2662  3          THEN
; 2625        2663  3              AP [NML$B_DTYPE] = DSC$K_DTYPE_B;
; 2626        2664  2          END;
; 2627        2665  2
; 2628        2666  2      RETURN 1;   ! Success
; 2629        2667  2
; 2630        2668  1      END;



                                             01FC 00000 LOOKUP_IDENTIFIER:
                                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8                ; 2486
                   58 00000000G   00   9E 00002              MOVAB   FOR$$SIGNAL_STO, R8                      ;
                   56         24   AC   D0 00009              MOVL    36(AP), NML_LIST                         ; 2545
                   55         10   AC   9E 0000D              MOVAB   16(AP), R5                               ; 2556
                   57         04   A6   3C 00011              MOVZWL  4(NML_LIST), I                           ;
                                   0E   11 00015              BRB     2$                                       ;
                   56              08   C0 00017 1$:          ADDL2   #8, NML_LIST                             ; 2555
                   54              66   D0 0001A              MOVL    (NML_LIST), R4                           ; 2556
                                0000V   30 0001D              BSBW    COMPARE_UPCASE                           ;
                   0D              50   E8 00020              BLBS    R0, 3$                                   ;
                                   57   D7 00023              DECL    I                                        ; 2553
                                   F0   12 00025 2$:          BNEQ    1$                                       ;
                                   55   DD 00027              PUSHL   R5                                       ; 2567
                                   13   DD 00029              PUSHL   #19                                      ;
                   68              02   FB 0002B              CALLS   #2, FOR$$SIGNAL_STO                      ;
                                   56   11 0002E              BRB     6$                                       ; 2569
              28   AC              66   D0 00030 3$:          MOVL    (NML_LIST), 40(AP)                       ; 2584
                   52         04   A6   D0 00034              MOVL    4(NML_LIST), DESC                        ; 2585
       50 3BBE001C 8F         02   A2   78 00038              ASHL    2(DESC), #1002307612, R0                 ; 2601
                                   3E   18 00041              BGEQ    5$                                       ;
                   01         03   A2   91 00043              CMPB    3(DESC), #1                              ; 2603
                                   06   13 00047              BEQL    4$                                       ;
                   04         03   A2   91 00049              CMPB    3(DESC), #4                              ;
                                   32   12 0004D              BNEQ    5$                                       ;
              2C   AC         04   A2   D0 0004F 4$:          MOVL    4(DESC), 44(AP)                          ; 2614
              34   AC         04   A2   D0 00054              MOVL    4(DESC), 52(AP)                          ; 2615
              38   AC              62   B0 00059              MOVW    (DESC), 56(AP)                           ; 2616
              44   AC         02   A2   90 0005D              MOVB    2(DESC), 68(AP)                          ; 2617
              3C   AC              52   D0 00062              MOVL    DESC, 60(AP)                             ; 2618
                   04         03   A2   91 00066              CMPB    3(DESC), #4                              ; 2620
                                   2A   12 0006A              BNEQ    8$                                       ;
         10   0A   A2              05   E1 0006C              BBC     #5, 10(DESC), 5$                         ; 2630
         0B   0A   A2              06   E1 00071              BBC     #6, 10(DESC), 5$                         ; 2631
                              0A   A2   95 00076              TSTB    10(DESC)                                 ; 2632
                                   06   18 00079              BGEQ    5$                                       ;
                   07         0B   A2   91 0007B              CMPB    11(DESC), #7                             ; 2633
                                   07   1B 0007F              BLEQU   7$                                       ;
                                   30   DD 00081 5$:          PUSHL   #48                                      ; 2636
```

```
                      68         01  FB 00083       CALLS   #1, FOR$$SIGNAL_STO          ;
                                 34  11 00086 6$:   BRB     11$                          ; 2637
        30  AC    2C  AC    0C   A2  C1 00088 7$:   ADDL3   12(DESC), 44(AP), 48(AP)     ; 2640
                  3A  AC    38   AC  B0 0008F       MOVW    56(AP), 58(AP)               ; 2641
                                 0D  11 00094       BRB     9$                           ; 2620
                      50    38   AC  3C 00096 8$:   MOVZWL  56(AP), R0                   ; 2645
                  30  AC    2C BC40  9E 0009A       MOVAB   @44(AP)[R0], 48(AP)          ;
                                 3A  AC  B4 000A0   CLRW    58(AP)                       ; 2646
                                 46  AC  94 000A3 9$: CLRB  70(AP)                       ; 2649
                  78  AC         01  D0 000A6       MOVL    #1, 120(AP)                  ; 2650
                  45  AC         0F  8A 000AA       BICB2   #15, 69(AP)                  ; 2654
                      02    44   AC  91 000AE       CMPB    68(AP), #2                   ; 2661
                                 04  12 000B2       BNEQ    10$                          ;
                  44  AC         06  90 000B4       MOVB    #6, 68(AP)                   ; 2663
                      50         01  D0 000B8 10$:  MOVL    #1, R0                       ; 2666
                                 04 000BB           RET                                  ;
                            50   D4 000BC 11$:      CLRL    R0                           ; 2668
                                 04 000BE           RET                                  ;
```

; Routine Size: 191 bytes, Routine Base: _FOR$CODE + 0631

; 2631 2669 1 !<BLF/PAGE>

```
1 %SBTTL 'SET_VALUE_IDENT - Mark that last token is supposed to be an identifier'
1 ROUTINE SET_VALUE_IDENT =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which is called when the character following
1 !       a value token indicates that the last token is supposed to be an
1 !       identifier.  It sets a flag in the parameter block which is checked
1 !       when the next identifier is needed.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = SET_VALUE_IDENT ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NML$V_VALUE_IDENT = 1
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     AP [NML$V_VALUE_IDENT] = 1;
2     RETURN 1;
2
1     END;
```

```
                         0000 00000 SET_VALUE_IDENT:
                                          .WORD   Save nothing                  ; 2671
     45   AC        04   88 00002         BISB2   #4, 69(AP)                    ; 2715
```

```
                         50          01  D0 00006        MOVL    #1, R0                                   ; 2716
                                         04 00009        RET                                              ; 2718

; Routine Size:  10 bytes,    Routine Base:  _FOR$CODE + 06F0

; 2682          2719  1 !<BLF/PAGE>
```

```
;  2684          2720  1 %SBTTL 'WAS_VALUE_IDENT - Lookup last value as an identifier'
;  2685          2721  1 ROUTINE WAS_VALUE_IDENT =
;  2686          2722  1
;  2687          2723  1 !++
;  2688          2724  1 ! FUNCTIONAL DESCRIPTION:
;  2689          2725  1 !
;  2690          2726  1 !       LIB$TPARSE action routine which is called when an identifier is needed.
;  2691          2727  1 !       If NML$V_VALUE_IDENT is 1 then the last value token is supposed to be
;  2692          2728  1 !       an identifier. Otherwise, 0 is returned.  The last value token, if it
;  2693          2729  1 !       could possibly be an identifier, was stored in NML$T_TOKEN.  We call
;  2694          2730  1 !       LOOKUP_IDENTIFIER to see if it is.  If the last token wasn't of type
;  2695          2731  1 !       REAL or LOGICAL or if the token length is zero, we fail.
;  2696          2732  1 !
;  2697          2733  1 ! CALLING SEQUENCE:
;  2698          2734  1 !
;  2699          2735  1 !       status = WAS_VALUE_IDENT ()
;  2700          2736  1 !
;  2701          2737  1 ! FORMAL PARAMETERS:
;  2702          2738  1 !
;  2703          2739  1 !       NONE
;  2704          2740  1 !
;  2705          2741  1 ! IMPLICIT INPUTS:
;  2706          2742  1 !
;  2707          2743  1 !       AP      Points to PARAM_BLOCK
;  2708          2744  1 !       NML$V_VALUE_IDENT
;  2709          2745  1 !       NML$T_TOKEN
;  2710          2746  1 !
;  2711          2747  1 ! IMPLICIT OUTPUTS:
;  2712          2748  1 !
;  2713          2749  1 !       See LOOKUP_IDENTIFIER
;  2714          2750  1 !
;  2715          2751  1 ! COMPLETION STATUS:
;  2716          2752  1 !
;  2717          2753  1 !       1 if LOOKUP_IDENTIFIER succeeds
;  2718          2754  1 !       0 if last token isn't an identifier
;  2719          2755  1 !
;  2720          2756  1 ! SIDE EFFECTS:
;  2721          2757  1 !
;  2722          2758  1 !       See LOOKUP_IDENTIFIER
;  2723          2759  1 !
;  2724          2760  1 !--
;  2725          2761  1
;  2726          2762  2     BEGIN
;  2727          2763  2
;  2728          2764  2     BUILTIN
;  2729          2765  2         AP;                 ! Argument pointer points to parameter block
;  2730          2766  2
;  2731          2767  2     MAP
;  2732          2768  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  2733          2769  2
;  2734          2770  2     LOCAL
;  2735          2771  2         TOKEN: REF VECTOR [, BYTE];
;  2736          2772  2
;  2737          2773  2     !+
;  2738          2774  2     ! If NML$V_VALUE_IDENT is 0, then fail.
;  2739          2775  2     !-
;  2740          2776  2
```

```
2       IF NOT .AP [NML$V_VALUE_IDENT]
2       THEN
2           RETURN 0;
2
2       !+
2       ! If last constant type is not REAL or LOGICAL or INTEGER or if token length
2       ! is zero, then we have a syntax error.
2       !-
2
2       IF NOT ONE_OF (.AP [NML$B_CONSTYPE], K_REAL, K_LOGICAL, K_INTEGER) OR
2           .AP [NML$T_TOKEN] EQL 0
2       THEN
3           BEGIN
3           !+
3           ! We reached this state by matching TPA$_LAMBDA just at the delimiter
3           ! that caused us to think that the last value token was really an
3           ! identifier.  TOKENPTR points to that delimiter and TOKENCNT is 0.
3           ! Increment TOKENCNT so that the delimiter will be in the error
3           ! message.
3           !-
3
3           AP [TPA$L_TOKENCNT] = .AP [TPA$L_TOKENCNT] + 1;
3           CALLG (.AP, SYNTAX_ERROR);
2           END;
2
2
2       !+
2       ! Construct token from NML$T_TOKEN.
2       !-
2
2       TOKEN = AP [NML$T_TOKEN];
2       AP [TPA$L_TOKENCNT] = .TOKEN [0];
2       AP [TPA$L_TOKENPTR] = TOKEN [1];
2       RETURN CALLG (.AP, LOOKUP_IDENTIFIER);
2
1       END;
```

```
                                      0000 00000 WAS_VALUE_IDENT:
                                                       .WORD   Save nothing                        : 2721
  2B       45   AC          02  E1 00002           BBC     #2, 69(AP), 3$                      : 2777
  50 70000000   8F     46   AC  78 00007           ASHL    70(AP), #1879048192, R0             : 2786
                            05  18 00010           BGEQ    1$                                  :
                       7C   AC  95 00012           TSTB    124(AP)                             : 2787
                            08  12 00015           BNEQ    2$                                  :
                       10   AC  D6 00017 1$:       INCL    16(AP)                              : 2798
         FE53   CF          6C  FA 0001A           CALLG   (AP), SYNTAX_ERROR                  : 2799
                50     7C   AC  9E 0001F 2$:       MOVAB   124(AP), TOKEN                      : 2807
           10   AC          60  9A 00023           MOVZBL  (TOKEN), 16(AP)                     : 2808
           14   AC     01   A0  9E 00027           MOVAB   1(R0), 20(AP)                       : 2809
         FF06   CF          6C  FA 0002C           CALLG   (AP), LOOKUP_IDENTIFIER             : 2810
                                04 00031           RET                                         :
                            50  D4 00032 3$:       CLRL    R0                                  : 2812
                                04 00034           RET                                         :
```

; Routine Size: 53 bytes, Routine Base: _FOR$CODE + 06FA

; 2777 2813 1 !<BLF/PAGE>

```
;  2779          2814  1 %SBTTL 'COMPARE_UPCASE - Compare strings upcased'
;  2780          2815  1 ROUTINE COMPARE_UPCASE (
;  2781          2816  1                         CSTRING_ADR,
;  2782          2817  1                         STRING2_DSC
;  2783          2818  1                         ) : JSB_COMPARE_UPCASE =
;  2784          2819  1
;  2785          2820  1 !++
;  2786          2821  1 ! FUNCTIONAL DESCRIPTION:
;  2787          2822  1 !
;  2788          2823  1 !       Compare two strings: the counted string whose address is CSTRING_ADR
;  2789          2824  1 !       and the string described by the descriptor STRING2_DSC.  The
;  2790          2825  1 !       STRING2_DSC string is upcased for the comparison; the CSTRING_ADR
;  2791          2826  1 !       string is assumed to be already upcased.
;  2792          2827  1 !
;  2793          2828  1 !       Comparison continues until a non-matching character is found or until
;  2794          2829  1 !       one of the strings is empty.  No blank-filling is done.
;  2795          2830  1 !
;  2796          2831  1 ! CALLING SEQUENCE:
;  2797          2832  1 !
;  2798          2833  1 !       matches = COMPARE_UPCASE (CSTRING_ADR, STRING2_DSC)
;  2799          2834  1 !
;  2800          2835  1 ! FORMAL PARAMETERS:
;  2801          2836  1 !
;  2802          2837  1 !       CSTRING_ADR     - The address of a counted string whose count is in the
;  2803          2838  1 !                         first byte.  Assumed to be uppercase.
;  2804          2839  1 !
;  2805          2840  1 !       STRING2_DSC     - The address of a string descriptor.  This string will
;  2806          2841  1 !                         be forced to upper case during the comparison.  The
;  2807          2842  1 !                         string itself is not modified.
;  2808          2843  1 !
;  2809          2844  1 ! IMPLICIT INPUTS:
;  2810          2845  1 !
;  2811          2846  1 !       NONE
;  2812          2847  1 !
;  2813          2848  1 ! IMPLICIT OUTPUTS:
;  2814          2849  1 !
;  2815          2850  1 !       NONE
;  2816          2851  1 !
;  2817          2852  1 ! FUNCTION VALUE:
;  2818          2853  1 !
;  2819          2854  1 !       1 if the strings are equal
;  2820          2855  1 !       0 otherwise
;  2821          2856  1 !
;  2822          2857  1 ! SIDE EFFECTS:
;  2823          2858  1 !
;  2824          2859  1 !       NONE
;  2825          2860  1 !
;  2826          2861  1 !--
;  2827          2862  1
;  2828          2863  2     BEGIN
;  2829          2864  2
;  2830          2865  2     MAP
;  2831          2866  2         CSTRING_ADR: REF VECTOR [, BYTE],
;  2832          2867  2         STRING2_DSC: REF BLOCK [, BYTE];
;  2833          2868  2
;  2834          2869  2     LOCAL
;  2835          2870  2         STRING2_ADR: REF VECTOR [, BYTE],
```

```
;  2836          2871  2          STRING1_LEN: WORD;
;  2837          2872  2
;  2838          2873  2      !+
;  2839          2874  2      ! Compare string lengths.  If they don't match, return failure.
;  2840          2875  2      !-
;  2841          2876  2
;  2842          2877  2      STRING1_LEN = .CSTRING_ADR [0];
;  2843          2878  2      IF .STRING1_LEN NEQU .STRING2_DSC [DSC$W_LENGTH]
;  2844          2879  2      THEN
;  2845          2880  2          RETURN 0;
;  2846          2881  2
;  2847          2882  2      !+
;  2848          2883  2      ! Compare strings for equality.  Lengths must match.
;  2849          2884  2      !-
;  2850          2885  2
;  2851          2886  2      STRING2_ADR = .STRING2_DSC [DSC$A_POINTER];
;  2852          2887  2      INCRU I FROM 1 TO .STRING1_LEN DO
;  2853          2888  3          BEGIN
;  2854          2889  3          IF .CSTRING_ADR [.I] NEQU
;  2855          2890  4              (
;  2856          2891  4              IF .STRING2_ADR [0] GEQU %C'a' AND .STRING2_ADR [0] LEQU %C'z'
;  2857          2892  4              THEN
;  2858          2893  5                  CH$RCHAR_A (STRING2_ADR) - (%C'a' - %C'A')
;  2859          2894  4              ELSE
;  2860          2895  4                  CH$RCHAR_A (STRING2_ADR)
;  2861          2896  4              )
;  2862          2897  3          THEN
;  2863          2898  3              RETURN 0;    ! Unequal character found
;  2864          2899  2          END;
;  2865          2900  2
;  2866          2901  2      !+
;  2867          2902  2      ! If we get here, then the match is successful.
;  2868          2903  2      !-
;  2869          2904  2
;  2870          2905  2      RETURN 1;
;  2871          2906  2
;  2872          2907  1      END;


                                50              64  9B 00000 COMPARE_UPCASE:
                                                            MOVZBW  (CSTRING_ADR), STRING1_LEN                  ;  2877
                                53              50  3C 00003         MOVZWL  STRING1_LEN, R3                     ;  2878
                                53              65  B1 00006         CMPW    (STRING2_DSC), R3                   ;
                                                33  12 00009         BNEQ    5$
                                51          04  A5  D0 0000B         MOVL    4(STRING2_DSC), STRING2_ADR         ;  2886
                                52              01  D0 0000F         MOVL    #1, I                               ;  2887
                                                21  11 00012         BRB     4$
                            61  8F              61  91 00014 1$:      CMPB    (STRING2_ADR), #97                  ;  2891
                                                0E  1F 00018         BLSSU   2$
                            7A  8F              61  91 0001A         CMPB    (STRING2_ADR), #122
                                                08  1A 0001E         BGTRU   2$
                                50              81  9A 00020         MOVZBL  (STRING2_ADR)+, R0                  ;  2893
                                50              20  C2 00023         SUBL2   #32, R0
                                                03  11 00026         BRB     3$
```

```
                                50              B1  9A 00028 2$:     MOVZBL  (STRING2_ADR)+, R0                              ; 2895
          50            6244  08              00  ED 0002B 3$:     CMPZV   #0, #8, (I)[CSTRING_ADR], R0                    ; 2890
                                                08  12 00031         BNEQ    5$                                              ;
                                                52  D6 00033         INCL    I                                               ; 2887
                                53              52  D1 00035 4$:     CMPL    I, R3                                           ;
                                                DA  1B 00038         BLEQU   1$                                              ;
                                50              01  D0 0C03A         MOVL    #1, R0                                          ; 2905
                                                    05 0003D         RSB                                                     ;
                                                50  D4 0003E 5$:     CLRL    R0                                              ; 2907
                                                    05 00040         RSB                                                     ;

; Routine Size:  65 bytes,    Routine Base:  _FOR$CODE + 072F
```

```
;  2874          2908  1 %SBTTL 'DUMP_NAMES - Respond to "?" inquiry'
;  2875          2909  1 ROUTINE DUMP_NAMES =
;  2876          2910  1
;  2877          2911  1 !++
;  2878          2912  1 ! FUNCTIONAL DESCRIPTION:
;  2879          2913  1 !
;  2880          2914  1 !       LIB$TPARSE action routine which is called when '?' is seen
;  2881          2915  1 !       in place of a variable.  If this file is a terminal on which we
;  2882          2916  1 !       have PUT access, call FOR$$DO_NML_OUTPUT to dump the group name
;  2883          2917  1 !       and variable names in the current namelist group.
;  2884          2918  1 !
;  2885          2919  1 ! CALLING SEQUENCE:
;  2886          2920  1 !
;  2887          2921  1 !       status = DUMP_NAMES ()
;  2888          2922  1 !
;  2889          2923  1 ! FORMAL PARAMETERS:
;  2890          2924  1 !
;  2891          2925  1 !       NONE
;  2892          2926  1 !
;  2893          2927  1 ! IMPLICIT INPUTS:
;  2894          2928  1 !
;  2895          2929  1 !       AP      Points to PARAM_BLOCK
;  2896          2930  1 !
;  2897          2931  1 ! IMPLICIT OUTPUTS:
;  2898          2932  1 !
;  2899          2933  1 !       NONE
;  2900          2934  1 !
;  2901          2935  1 ! COMPLETION STATUS:
;  2902          2936  1 !
;  2903          2937  1 !       1
;  2904          2938  1 !
;  2905          2939  1 ! SIDE EFFECTS:
;  2906          2940  1 !
;  2907          2941  1 !       May list namelist group on terminal.
;  2908          2942  1 !
;  2909          2943  1 !--
;  2910          2944  1
;  2911          2945  2     BEGIN
;  2912          2946  2
;  2913          2947  2     BUILTIN
;  2914          2948  2         AP;                 ! Argument pointer points to parameter block
;  2915          2949  2
;  2916          2950  2     MAP
;  2917          2951  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  2918          2952  2
;  2919          2953  2     GLOBAL REGISTER
;  2920          2954  2         CCB = 11: REF $FOR$CCB_DECL;
;  2921          2955  2
;  2922          2956  2     CCB = .AP [NML$A_CCB];      ! Load CCB register
;  2923          2957  2
;  2924          2958  2     !+
;  2925          2959  2     ! If we are on a terminal with PUT access, list the namelist group.
;  2926          2960  2     !-
;  2927          2961  2
;  2928          2962  3         BEGIN
;  2929          2963  3         BIND
;  2930          2964  3             FAB = CCB: REF $FOR$FAB_CCB_STRUCT,
```

```
; 2931          2965  3               FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];
; 2932          2966  3
; 2933          2967  3           IF .FAB_DEV [DEV$V_TRM] AND .FAB [FAB$V_PUT]
; 2934          2968  3           THEN
; 2935          2969  4               BEGIN
; 2936          2970  4               FOR$$REC_WSNO ();   ! Start output record
; 2937          2971  4               FOR$$DO_NML_OUTPUT (1);     ! Dump names only
; 2938          2972  3               END;
; 2939          2973  2           END;
; 2940          2974  2
; 2941          2975  2       RETURN 1;
; 2942          2976  2
; 2943          2977  1       END;


                                        083C 00000 DUMP_NAMES:
                                                         .WORD   Save R2,R3,R4,R5,R11                     ; 2909
                         5B        40   AC  D0 00002     MOVL    64(AP), CCB                              ; 2956
                         50      0084   CB  9E 00006     MOVAB   132(R11), R0                             ; 2965
              13         60             02  E1 0000B     BBC     #2, (R0), 1$                             ; 2967
                         0F        5A   AB  E9 0000F     BLBC    90(FAB), 1$                              ;
                            00000000G   00  16 00013     JSB     FOR$$REC_WSNO                            ; 2970
                                        01  DD 00019     PUSHL   #1                                       ; 2971
                  00000000G  00         01  FB 0001B     CALLS   #1, FOR$$DO_NML_OUTPUT                   ;
                             50         01  D0 00022 1$: MOVL    #1, R0                                   ; 2975
                                            04 00025     RET                                              ; 2977

; Routine Size:  38 bytes,    Routine Base:  _FOR$CODE + 0770
```

```
;  2945          2978  1 %SBTTL 'DUMP_VALUES - Respond to "=?" inquiry'
;  2946          2979  1 ROUTINE DUMP_VALUES =
;  2947          2980  1
;  2948          2981  1 !++
;  2949          2982  1 ! FUNCTIONAL DESCRIPTION:
;  2950          2983  1 !
;  2951          2984  1 !       LIB$TPARSE action routine which is called when '=?' is seen
;  2952          2985  1 !       in place of a variable.  If this file is a terminal on which we
;  2953          2986  1 !       have PUT access, call FOR$$DO_NML_OUTPUT to dump the group name
;  2954          2987  1 !       and variable names and values in the current namelist group.
;  2955          2988  1 !
;  2956          2989  1 ! CALLING SEQUENCE:
;  2957          2990  1 !
;  2958          2991  1 !       status = DUMP_VALUES ()
;  2959          2992  1 !
;  2960          2993  1 ! FORMAL PARAMETERS:
;  2961          2994  1 !
;  2962          2995  1 !       NONE
;  2963          2996  1 !
;  2964          2997  1 ! IMPLICIT INPUTS:
;  2965          2998  1 !
;  2966          2999  1 !       AP      Points to PARAM_BLOCK
;  2967          3000  1 !
;  2968          3001  1 ! IMPLICIT OUTPUTS:
;  2969          3002  1 !
;  2970          3003  1 !       NONE
;  2971          3004  1 !
;  2972          3005  1 ! COMPLETION STATUS:
;  2973          3006  1 !
;  2974          3007  1 !       1
;  2975          3008  1 !
;  2976          3009  1 ! SIDE EFFECTS:
;  2977          3010  1 !
;  2978          3011  1 !       May list namelist group on terminal.
;  2979          3012  1 !
;  2980          3013  1 !--
;  2981          3014  1
;  2982          3015  2     BEGIN
;  2983          3016  2
;  2984          3017  2     BUILTIN
;  2985          3018  2         AP;                 ! Argument pointer points to parameter block
;  2986          3019  2
;  2987          3020  2     MAP
;  2988          3021  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
;  2989          3022  2
;  2990          3023  2     GLOBAL REGISTER
;  2991          3024  2         CCB = 11: REF $FOR$CCB_DECL;
;  2992          3025  2
;  2993          3026  2     CCB = .AP [NML$A_CCB];      ! Load CCB register
;  2994          3027  2
;  2995          3028  2     !+
;  2996          3029  2     ! If we are on a terminal with PUT access, list the namelist group.
;  2997          3030  2     !-
;  2998          3031  2
;  2999          3032  3         BEGIN
;  3000          3033  3         BIND
;  3001          3034  3             FAB = CCB: REF $FOR$FAB_CCB_STRUCT,
```

```
; 3002          3035  3                FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];
; 3003          3036  3
; 3004          3037  3            IF .FAB_DEV [DEV$V_TRM] AND .FAB [FAB$V_PUT]
; 3005          3038  3            THEN
; 3006          3039  4                BEGIN
; 3007          3040  4                FOR$$REC_WSNO ();    ! Start output record
; 3008          3041  4                FOR$$DO_NML_OUTPUT (0);      ! Dump names and values
; 3009          3042  3                END;
; 3010          3043  2            END;
; 3011          3044  2
; 3012          3045  2        RETURN 1;
; 3013          3046  2
; 3014          3047  1        END;



                                          083C 00000 DUMP_VALUES:
                                                          .WORD    Save R2,R3,R4,R5,R11                        ; 2979
                        5B         40  AC  D0 00002      MOVL     64(AP), CCB                                 ; 3026
                        50       0084  CB  9E 00006      MOVAB    132(R11), R0                                ; 3035
              13        60             02  E1 0000B      BBC      #2, (R0), 1$                                ; 3037
                        0F         5A  AB  E9 0000F      BLBC     90(FAB), 1$                                 ;
                            00000000G  00  16 00013      JSB      FOR$$REC_WSNO                               ; 3040
                                       7E  D4 00019      CLRL     -(SP)                                       ; 3041
                 00000000G  00         01  FB 0001B      CALLS    #1, FOR$$DO_NML_OUTPUT                      ;
                            50         01  D0 00022 1$:  MOVL     #1, R0                                      ; 3045
                                           04 00025      RET                                                  ; 3047

; Routine Size:  38 bytes,    Routine Base:  _FOR$CODE + 0796
```

```
; 3016          3048  1 END                                                 ! End of module FOR$$NML_TABLES
; 3017          3049  1
; 3018          3050  0 ELUDOM
```

;   PSECT SUMMARY

Name	Bytes	Attributes
_LIB$KEY0$	0	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(1)
_LIB$STATE$	1050	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(1)
_FOR$CODE	1980	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

;   Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
_255DUA28:[SYSLIB]STARLET.L32;1	9776	37	0	581	00:01.1
_255DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	216	30	52	00:00.5
_255DUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1
_255DUA28:[SYSLIB]TPAMAC.L32;1	42	27	64	14	00:00.1

;   COMMAND QUALIFIERS

```
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:FORNMLTAB/OBJ=OBJ$:FORNMLTAB MSRC$:FORNMLTAB/UPDATE=(ENH$:FORNMLTAB
;       )

; Size:          1980 code + 1050 data bytes
; Run Time:         01:58.7
; Elapsed Time:     04:10.6
; Lines/CPU Min:     1541
; Lexemes/CPU-Min: 73012
; Memory Used:  417 pages
; Compilation Complete
```

FORUNDUND
LIS

FORMSG
LIS

FORIOBEG
LIS

FORIOEND
LIS

FORLEX
LIS

FORNMLTAB
LIS

FORINQUIR
LIS

FORIOELEM
LIS

FORJDATE
LIS

FORLIB
LIS

FORRAB
LIS

FORREADDF
LIS

FOROPNKEY
LIS

FORPAUSE
LIS

FORRANDOM
LIS

FOROPEN
LIS

FOROPENDE
LIS

FORREADDO
LIS